Guide to VAXclusters

Order Number: AA–Y513A–TE

September 1984

The *Guide to VAXclusters* describes the procedures for setting up and managing a VAXcluster. The information presented in this document describes VAXcluster functions for VAX/VMS Version 4.0.

Revision/Update Information:	This is a new manual.
Software Version:	VAX/VMS Version 4.0

digital equipment corporation
maynard, massachusetts

September 1984

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital

ZK–2538

This document was prepared using an in-house documentation production system. All page composition and make-up was performed by TEX, the typesetting system developed by Donald E. Knuth at Stanford University. TEX is a registered trademark of the American Mathematical Society.

Contents

EXAMPLES

FIGURES

TABLES

Preface

Intended Audience

This document is intended for users responsible for setting up and managing a VAXcluster. To use this document as a guide to cluster management, you must have a thorough understanding of the VAX/VMS operating system, particularly system management.

Structure of This Document

The *Guide to VAXclusters* contains five chapters and four appendixes.

Chapter 1 provides an overview of the VAXcluster and its hardware and software components.

Chapter 2 explains how to coordinate various system files and system command procedures to prepare the VAXcluster operating environment before forming a VAXcluster.

Chapter 3 describes cluster queue management concepts and provides sample procedures for setting up cluster queues.

Chapter 4 describes cluster disks and provides sample command procedures for setting up and mounting them.

Chapter 5 explains how to form the VAXcluster once the necessary preparations are made. This chapter includes a discussion of VAXcluster Connection Management, descriptions of required SYSGEN parameters, and procedures for booting, managing, and shutting down the cluster.

Appendix A contains guidelines for building a cluster-common user authorization file for an upgraded system.

Appendix B contains a list of the connection manager messages that are displayed at the operator's terminal.

Appendix C contains a procedure for booting from a common system disk.

Appendix D contains examples of command files used to start up a cluster.

Associated Documents

The *Guide to VAXclusters* describes cluster management concepts and procedures and is not a one-volume reference source for VAXcluster information. The utilities and commands used in this document are described in separate sections of both the *VAX/VMS Utilities Reference Volume* and the *VAX/VMS DCL Dictionary*.

The *Guide to VAX/VMS System Management and Daily Operations* is a prerequisite, especially for those using this document as a guide to cluster management.

New and Changed Features

The *Guide to VAXclusters* is a new document. It describes VAXcluster functions for VAX/VMS Version 4.0.

1 Cluster Overview

The VAXcluster is a highly integrated organization of VAX/VMS systems that communicate over a high-speed communications path. Like a single-node VAX/VMS system, the VAXcluster organization provides a single security and management domain.

The VAXcluster does not require its own operating system. Rather, the VAX/VMS operating system is extended to support VAXclusters. As a result, VAXclusters have all the functions of single node VAX/VMS systems, plus the ability to share CPU resources, queues, and disk storage. It is this ability to share resources under a single security and management perimeter that distinguishes VAXclusters from tightly coupled multiprocessor systems and loosely coupled networks.

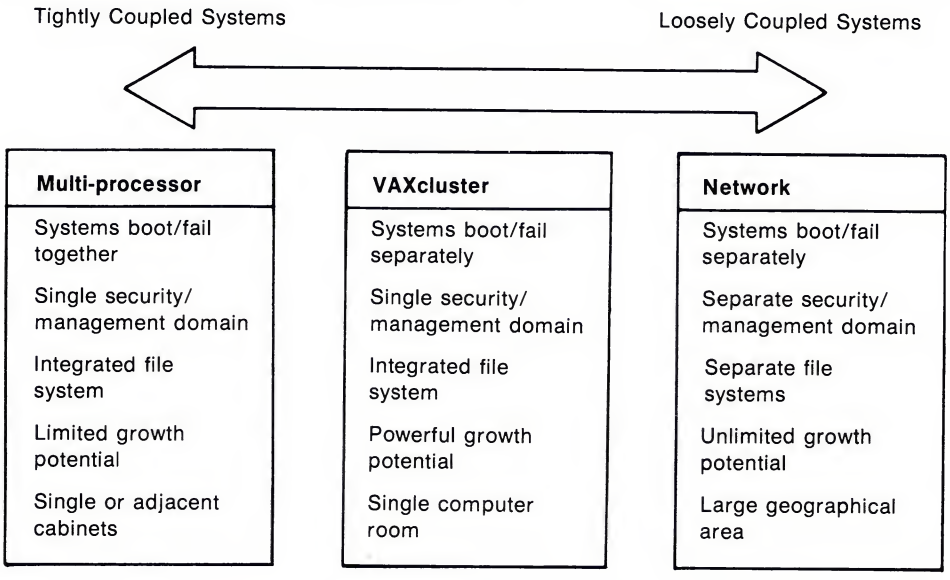
VAXclusters are best understood when compared to tightly coupled multiprocessor systems and loosely coupled networks. Figure 1–1 illustrates how VAXclusters fit into the range of multiprocessor architectures.

1.1 Hardware

Figure 1–2 shows the hardware components of a VAXcluster.

The basic components of a VAXcluster are listed below. Unlike the other components, the HSC50 is not required; however, the HSC50 provides additional capability for the cluster. Note that the total of VAX processors and HSC50s in a VAXcluster must not exceed 16.

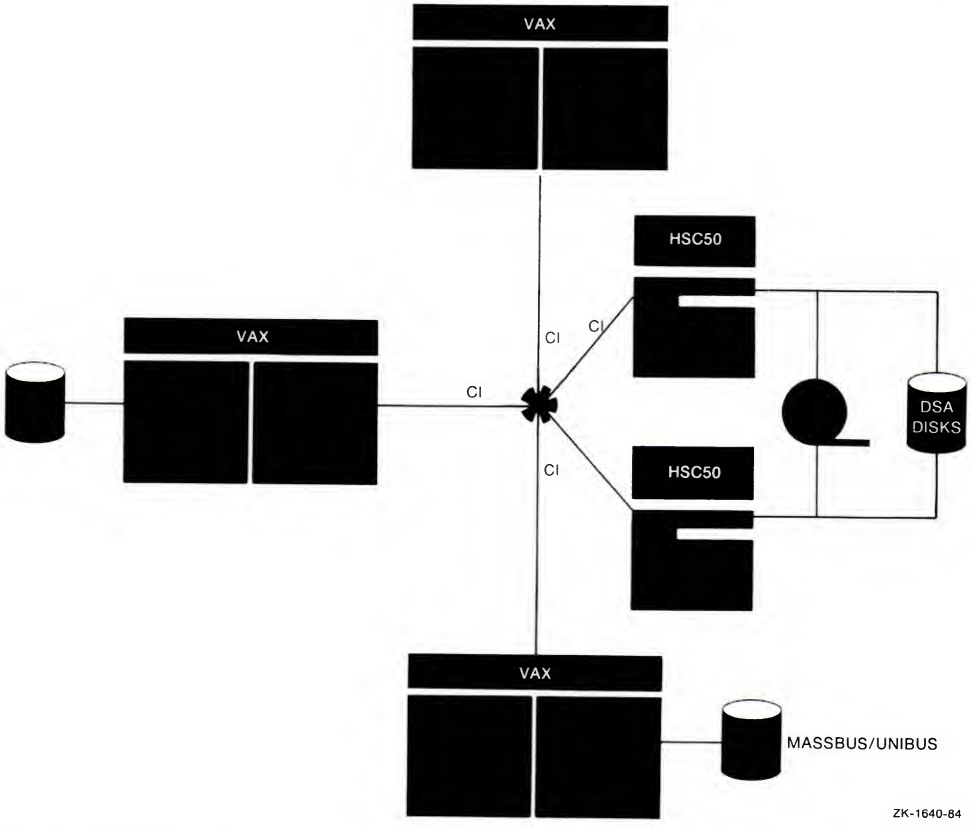
Figure 1–1 VAXclusters and Other Multiprocessors



Component	Function
VAX processor	A VAX–11/780, VAX–11/782, VAX–11/785, or VAX–11/750 processor running the VAX/VMS operating system. Any one of these processors is considered an *active node* in the cluster. The VAXcluster supports up to 16 active nodes.
CI	The computer interconnect (CI) is a high-speed, dual-path, bus that connects processor nodes (VAX) and intelligent I/O subsystems (HSC50) in a computer room environment. There can be only one CI on each VAXcluster.

Component	Function
Controller	The CI780 is a microcoded, intelligent controller that connects VAX–11 /780, VAX–11/782, or VAX–11/785 processor to the CI. The CI750 is a microded, intelligent controller that connects the VAX–11/750 processor to the CI. Each interface connects to the CI bus, which consists of two transmit and two receive cables. Under normal operating conditions, both sets of cables are available to meet traffic demands. If one path becomes inoperative, then all traffic uses the remaining path. The VAX/VMS operating system periodically tests a failed path. As soon as a failed path becomes available, it will automatically be used for normal traffic.
Star coupler	The star coupler is the common connection point for all nodes connected to the CI. As in the case of the CI bus, the star coupler is dual-pathed and contains separate components for each path. The star coupler connects all CI cables from the individual nodes, creating a radial or "star" arrangement that has a maximum radius of 45 meters. It supports the physical connection or disconnection of nodes during normal cluster operations, without affecting the rest of the cluster.
HSC50	The Hierarchical Storage Controller (HSC50) is a self-contained, intelligent, mass storage subsystem that enables VAXcluster nodes to share DIGITAL Standard Architecture (DSA) disks. Because the HSC50 is an intelligent controller, it optimizes physical disk operations. The HSC50 is considered a *passive node*. The VAXcluster supports up to 15 HSC50s.

Figure 1–2 VAXcluster Hardware Components



1.2 Software

The software used to implement VAXclusters includes the following six components:

- System Communication Services
- Connection Manager
- Distributed File System
- Distributed Lock Manager
- Distributed Job Controller
- MSCP Server

The *system communication services* (SCS) is the software layer that implements internode communication, according to DIGITAL's System Communication Architecture.

A list of the SCS related SYSGEN parameters is included in Chapter 5.

The *connection manager* is the software layer that dynamically defines and coordinates the VAXcluster. The connection manager uses the system communication services and provides an acknowledged message delivery service for higher VMS software layers. The connection manager also conducts cluster state changes—that is, nodes joining or leaving the cluster.

The connection manager is described in Chapter 5. A list of the connection manager messages displayed at the operator's terminal is contained in Appendix B.

The *distributed file system* allows all VAX processors to share disk mass storage, whether the disk is connected to an HSC50 or to a VAX processor. A local disk may be made available to the entire cluster, and all cluster available disks appear as if they are local to every VAX processor.

The distributed file system and VAX Record Management Services (RMS) provide the same access to disks and files cluster-wide that is provided on a single node system. RMS files may be shared cluster-wide to the record level.

The *distributed lock manager* is used by the file system, RMS, job controller, device allocation, and other cluster facilities for synchronization. It is available to users to develop cluster applications. The distributed lock manager implements the $ENQ and $DEQ system services to provide cluster-wide synchronization of access to resources.

The lock manager provides a mechanism to lock and unlock resource names. It also provides a queuing mechanism so that processes can be put into a wait state until a particular resource is available. As a result, cooperating processes can synchronize their access to shared objects such as files or records.

If a VAX processor in the cluster should fail, all locks held by the failed VAX processor are released. This allows processing to continue on the remaining VAX processors.

The distributed lock manager also supports cluster-wide deadlock detection.

The *distributed job controller* makes queues available cluster wide. A VAXcluster system operates with a common set of batch and print queues for the entire cluster. Users can submit jobs to any queue within the cluster provided that the necessary mass storage volumes and peripheral devices are accessible to the system on which the job executes.

You can also set up generic batch queues that distribute batch processing workloads between cluster nodes. Cluster queues are described in Chapter 3.

The *Mass Storage Control Protocol (MSCP) server* implements the MSCP protocol, which is used to communicate with a controller for DSA disks, such as HSC50s. The MSCP server implements this protocol on a VAX processor, submits the I/O requests to local UNIBUS, MASSBUS, and UDA disks, and accepts the I/O request from any node in the cluster. In this way, the MSCP server makes locally connected disks available to all nodes in the cluster.

The MSCP server is described in Chapter 4.

2 Preparing the VAXcluster Environment

Before you attempt to form a VAXcluster, you should prepare the cluster operating environment. The preparation you need depends on the hardware configuration of your VAXcluster and the type of environment your site requires. This chapter describes the types of cluster environments you can set up and explains the steps you should take to prepare them.

2.1 The Cluster Operating Environment

The operating environment you may prepare for your cluster ranges from homogeneous to heterogeneous. The type you choose depends mainly on the processing needs of your site.

In a *homogeneous cluster* the operating environment is identical on each member node, either because the nodes are run from the same system files or because they share a few selected system files that control the user environment. The nodes are set up with identical user accounts, the same known images are installed, the same logical names are defined, and mass storage devices and queues are shared. In effect, users in a homogeneous cluster can log in to any node and work in the same operating environment.

In a *heterogeneous cluster* the environment on each node is unique. Users work in different operating environments, environments that are specific to the node they are logged in to. A heterogeneous cluster is effective when you want the member nodes to serve specialized needs, but you need to share data among them. The less you coordinate the environments of the individual cluster nodes (that is, the less you make them the same), the more heterogeneous your cluster will be.

You can also set up a cluster that is somewhere between homogeneous and heterogeneous. For example, you might want to set up a three-node cluster, in which the time-sharing environments on two nodes are the same, and the third node is set up exclusively for batch processing of large inventory jobs. In this case, the time-sharing nodes are set up homogeneously,

sharing users, queues, and access to mass storage devices, while the third node runs in its own restricted environment.

As you can imagine, the number of possible cluster environments that you can create is countless. For this reason, this chapter concentrates on the steps necessary in preparing a homogeneous cluster. Approaches for preparing a more heterogeneous cluster are also described, but are presented as general guidelines. The degree to which you follow these guidelines will determine how heterogeneous a cluster you will create.

2.2 Installing the New Operating System

The approach you use when installing or upgrading the operating system depends on the type of cluster environment you want to create. You must perform the installation or upgrade once for each system disk in the cluster but, since several nodes may be run from the same system disk, you do not necessarily have to perform the installation or upgrade on each cluster node.

Note: All active cluster nodes must run the same major and minor versions of the VAX/VMS operating system. Nodes running the Version 4.0 field system cannot coexist in the cluster with nodes running the Version 3.7 system or earlier. Similarly, when it is time to upgrade from the Version 4.0 to the Version 4.1 system, all nodes must be upgraded simultaneously or leave the cluster.

The following three subsections present guidelines for installing a VAXcluster on systems using

- A common system disk
- Individual system disks
- A combination of common and individual system disks

For more detailed information on installation procedures, see the description in the *Guide to VAX/VMS Software Installation.*

2.2.1 Using a Common System Disk

You will be asked, during the installation or upgrade, whether you want to set up a common system disk—that is, a cluster in which each node is run from the same system disk. Note that if you select this option, you must use an HSC50 disk as the system disk.

2.2.1.1 Creating a Common System Disk

To set up a cluster in which each node runs from a common system disk, you begin by installing or upgrading the system on one node. After the upgrade has completed and the system is up and running, invoke the MAKEROOT.COM command procedure to create system roots for nodes other than the first node of the cluster. To invoke MAKEROOT, enter

```
$ @SYS$MANAGER:MAKEROOT
```

Note that MAKEROOT will abort if your system disk is not a common system disk, or if the SYSGEN parameter VAXCLUSTER is 0.

When MAKEROOT executes, it requests the name of the new system root. You enter the root name, using the form SYSx where x is a hexadecimal digit in the range 1 through 9 and A through D (for example, SYS1 or SYSA). Note that system roots SYSE and SYSF are reserved for other system functions.

You may not specify the root of the currently running system (usually SYS0). If you specify any other existing system root, you will be asked if you wish to continue—that is, if you want to modify the existing system root. If you do want to modify the existing system root, MAKEROOT will delete the following files from it:

- [SYSEXE]MODPARAMS.DAT
- [SYSEXE]VAXVMSSYS.PAR
- [SYSMGR]VMSIMAGES.DAT.

If a SYSCOMMON directory exists in the directory tree, it will be removed.

Note: MAKEROOT does not check the format of the existing system root. DIGITAL recommends you delete all the files

in the directory tree, and the directory tree itself, or choose a different root.

MAKEROOT will then ask for the new nodename (which cannot consist of more than six characters) and its SCSSYSTEMID. After you provide this information, MAKEROOT will prompt for the size of new root's page file and swap file. The values you provide are subsequently used by AUTOGEN.

Finally, MAKEROOT creates the new directory tree, the page file and the swap file. It also generates a VAXVMSSYS.PAR file for the new root using the SYSGEN parameters of the currently running node as the basis for the new node.

When MAKEROOT completes this operation, it displays the following message:

```
You must now boot the target node into the newly created root.
Use a conversational bootstrap, as you must change the startup
command procedure. Use the SYSBOOT commands:

SET /STARTUP SYS$SYSTEM:STARTUP.COM
SET STARTUP_P1 "MIN"
CONTINUE
```

Following the system boot, log in and invoke AUTOGEN using this command:

```
$ @SYS$UPDATE:AUTOGEN GETDATA REBOOT
```

For more detailed information on the procedures for booting from an HSC50 common system disk, see the description in Appendix C.

2.2.1.2 Locating Files on a Common System Disk

Most operating system files will automatically be placed in a common directory tree, [V4COMMON]. Those files that are not (which include PAGEFILE.SYS, SWAPFILE.SYS, SYSDUMP.DMP, VAXVMSSYS.PAR, VMSIMAGES.DAT, log files, and node-specific DECnet files) can be found in the node-specific directory trees [SYS0], [SYS1], and so on.

If a SYSCOMMON subdirectory is found in the system-specific [SYSn] directory tree at boot time, the system logical name SYS$SYSROOT is redefined to be the search list consisting of the local node directory tree and the [V4COMMON] directory tree.

When you install optional software products using VMSINSTAL, and a SYSCOMMON directory is found in the node-specific directory tree, VMSINSTAL will install into the [V4COMMON] tree. You can override this default, however, by specifying the option to install into an alternate root.

Optional products that use VMSUPDATE may not be installed on a common system disk. If a layered product you need is not available for installation with VMSINSTAL, please contact DIGITAL.

2.2.2 Using Individual System Disks

To set up a cluster in which each node uses its own system disk, install or upgrade the system on each cluster node and boot each node to run as a *single system* (a system that is not a cluster member).

To boot a node as a single system, DIGITAL recommends that you shut down all other active nodes connected to the CI before booting the node. Doing so prevents data corruption by eliminating the possibility of two processors accessing data in an uncoordinated manner. (See the discussion of cluster partitioning in Chapter 5.) Once a node is running as a single system, you can make the necessary adjustments outlined in this chapter to prepare the node to run as part of the cluster.

Note that you cannot use a dual-ported MASSBUS disk as a system disk. Note also that if you use an MSCP-served disk as a system disk, that disk must be local (connected) to the node from which it is booting.

2.2.3 Using a Combination of Common and Individual System Disks

You may want to set up a cluster that has a combination of one or more common system disks and one or more individual system disks.

Again, you must do the installation or upgrade once for each system disk. For example, if your cluster consists of ten nodes and four of them share one common system disk, four other nodes share a second common system disk, and the remaining

two nodes each have their own system disk, you would have to install or upgrade the system four times.

2.3 Coordinating System Command Procedures

You should coordinate the system command procedures, SYSTARTUP.COM and SYLOGIN.COM, according to the type of cluster operating environment you want to prepare. For a homogeneous cluster, these procedures should perform the same system startup and login functions for each cluster node. For a more heterogeneous cluster, you may want some SYSTARTUP and SYLOGIN commands to remain specific to certain nodes.

Once you have created the common SYSTARTUP and SYLOGIN command procedures, you can set up each of them as a common file on a shared disk or as separate duplicate files.

Using either approach, you can include a command in the node-specific SYSTARTUP.COM that will invoke the common SYSTARTUP procedure. In a homogeneous cluster, the node-specific version of SYSTARTUP.COM for each node invokes a common startup procedure, named for example, SYSTARTUP_COMMON.COM. Thus, each SYSTARTUP.COM procedure on each node would include a command similar to the following:

```
@DISK$:[SYSMGR]STARTUP_COMMON.COM
```

An example of a common startup command file is included in Appendix D. Examples of two node-specific startup command files are also included.

Certain startup functions, even in a homogeneous cluster, are node specific. Therefore, you should include commands in the node-specific SYSTARTUP.COM procedure on each node to do the following:

- Set up dual-ported disks
- Load device drivers
- Set up terminals
- Invoke the common SYSTARTUP command procedure

If the common SYSTARTUP procedure is on a local disk, the node-specific procedure must set up the local disk as a shared disk before invoking the common procedure. If the procedure is not on the system disk, the disk on which it resides must be mounted before the procedure can be invoked.

Alternatively, you could set up duplicate copies of the common procedure on a separate volume on each cluster node. To set up a common SYLOGIN procedure, define the logical name SYS$SYLOGIN on each cluster node to be the full file specification of the procedure. If the common SYLOGIN file is on a shared disk, you can include the command that defines SYS$SYLOGIN in the common SYSTARTUP procedure. If the cluster nodes use separate duplicate copies of SYLOGIN, you should include the definition in the node-specific SYSTARTUP.COM for each node.

For example, the following command defines SYS$SYLOGIN to be the common file [SYSMGR]SYLOGIN on the shared disk WORK1.

```
$ DEFINE/SYSTEM/EXEC SYS$SYLOGIN WORK1:[SYSMGR]SYLOGIN
```

The next two sections present guidelines for using common and node-specific command procedures to build a cluster environment. You can use these procedures to tailor your cluster to various degrees of homogeneity or heterogeneity, depending on your particular needs.

2.3.1 Building Common Command Procedures

The first step in preparing a homogeneous cluster is to build common system startup and login command procedures. To build these procedures for a cluster in which existing nodes are to be merged, you should compare both the SYSTARTUP and SYLOGIN command procedures on each node and make any adjustments required. For example, you can compare the SYSTARTUP.COM procedure from each node and include commands that define the same logical names in the common startup command procedure.

If your cluster consists of nodes that were upgraded to Version 4.0 from a previous version of VAX/VMS, compare the existing SYSTARTUP and SYLOGIN procedures from each node and

decide which elements you want common for all nodes. You should include these common elements in a common command procedure file that will be executed by each node.

An easy method of comparing the existing procedures and creating common versions is to log into each cluster node (in the single-system environment) and print the existing SYSTARTUP and SYLOGIN command procedure files. You can then use the file listings to compare the procedures. After you have chosen which commands to make common, you can build the common procedures on one of the cluster nodes.

The strategy for clusters being formed from newly installed VAX/VMS systems is basically the same as that used for clusters using upgraded VAX/VMS systems—that is, include common elements in a common command procedure file. With newly installed systems, however, the SYSTARTUP and SYLOGIN command procedure files are empty, so you build the common procedures from scratch.

To build a common SYSTARTUP command procedure, create a file named, for example, SYSTARTUP_COMMON.COM, and include the commands that you want common to all nodes. You should decide which of the following elements you want to include in the common SYSTARTUP procedure:

- Commands that install images.
- Commands that define logical names; for example, the logical name that refers to the location of SYLOGIN.COM.
- Commands that set up queues.
- Commands that set up and mount physically accessible mass storage devices.
- Commands that perform any other common site-specific startup functions. See the *Guide to VAX/VMS System Management and Daily Operations* for more information on startup command procedures.

In a common SYSTARTUP command procedure, the execution sequence of commands that set up queues and mount cluster-accessible devices is node dependent. Therefore, you must include conditional DCL commands to control how these commands are executed.

You can include commands that set up queues and mount cluster-accessible devices as part of the common SYSTARTUP procedure or as separate command procedures, such as STARTQUE_COMMON or MOUNT_COMMON that are invoked by the common procedure. Sample procedures for setting up queues and mounting cluster-accessible volumes are described in Chapter 3 and Chapter 4, respectively.

Note: The job-controller queue file, JBCSYSQUE.DAT, must be set up as a common file on a shared disk, accessible to all the nodes sharing queues. If you intend to set up common procedures such as SYSTARTUP_COMMON.COM or STARTQ_COMMON.COM as a single common file on a shared disk volume, it is a good idea to locate these files on the same shared volume containing JBCSYSQUE.DAT.

To build a common SYLOGIN.COM command procedure, include any of the following commands in a common SYLOGIN command procedure file:

- Commands that define symbols
- Commands that perform other site-specific login functions

2.3.2 Using Node-Specific System Command Procedures

To create a more heterogeneous environment, include elements that you want to remain unique to a node, such as commands to define node-specific logical names, in the node-specific version of SYSTARTUP.COM and SYLOGIN.COM for that node.

For example, consider a three-node cluster consisting of nodes MOE, LARRY, and CURLY. The time-sharing environments on nodes MOE and LARRY are the same. CURLY is set up for specific turnkey accounts. In this case, you could create common startup and login command procedures for nodes MOE and LARRY that set up identical environments on these nodes. The command procedures for node CURLY, however, would be different, set up specifically for CURLY's turnkey environment.

In a totally heterogeneous cluster, common system procedures are not necessary because the environment on each cluster node is different. So, to create a totally heterogeneous cluster, keep your current node-specific command procedures intact.

2.4 Coordinating System Files to Build a Common User Environment

In a homogeneous cluster, users can log in to any cluster node and work in the same operating environment because the system files that control these user functions are identical. You make them identical either by running all the cluster nodes from a separate copy of the VAX/VMS operating system, or from a common copy shared by all nodes.

Copies of the operating system may reside on individual system disks or in separate roots on a common system disk. The next two sections describe how to coordinate files in each type of system disk configuration in the cluster.

2.4.1 Coordinating Files on a Common System Disk

If your cluster is running from a common system disk, most operating system and layered product files are stored in a common root directory.

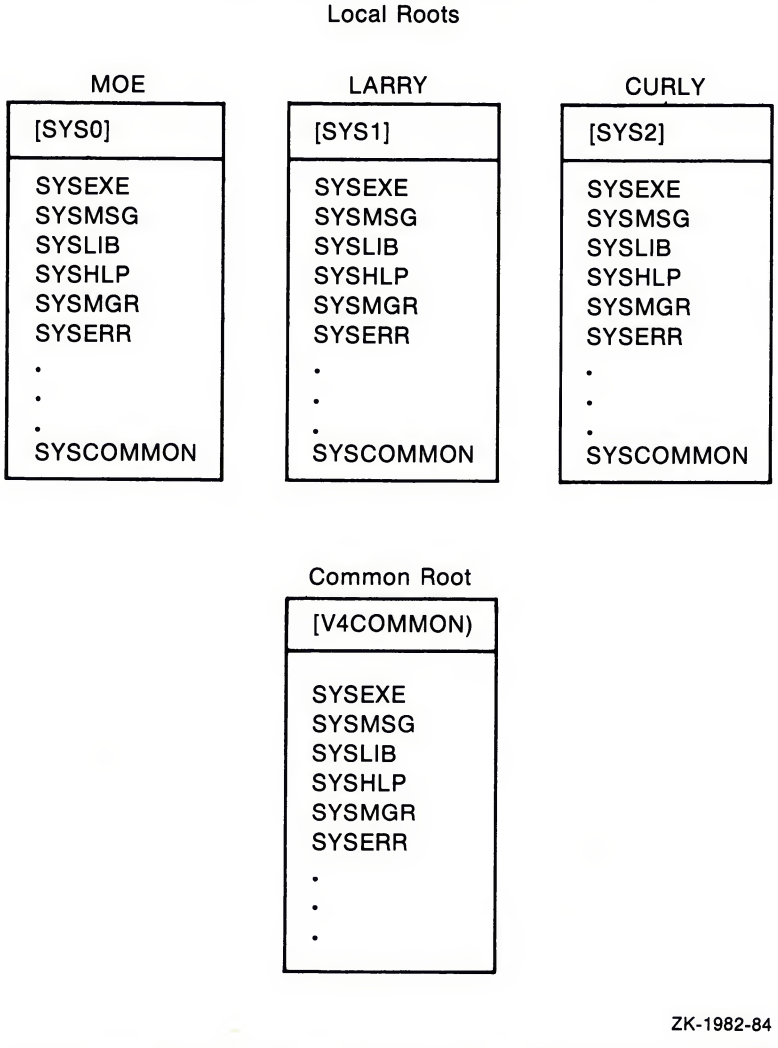
The logical name SYS$SYSROOT is defined as a search list that points to a local root first (SYS$SPECIFIC), and then to the common root (SYS$COMMON). Thus, the logical names for the system directories (SYS$SYSTEM, SYS$LIBRARY, SYS$MANAGER, and so forth) point to two directories: a local root (for example, SYS$SPECIFIC:[SYSEXE]) and a common root (for example, SYS$COMMON:[SYSEXE]).

Note: Some layered products may not support common system files. Also note that layered products installed in the common system disk *must* be licensed for all nodes booting from that disk.

The entire directory structure—that is, the common root plus each node's local root—is stored on the same HSC50 disk. To set up this structure, install or upgrade the Version 4.0 system on one node, then follow the guidelines in Section 2.2.1.1 to create local roots for the other nodes. Once a local root is set

up for a particular node, you can use the common system disk to boot that node and run AUTOGEN.

Figure 2–1 Directory Structure on Common System Disk



ZK-1982-84

Figure 2–1 illustrates the directory structure set up for three nodes (MOE, LARRY, and CURLY) that are run from a common system disk. The disk's master file directory (MFD) contains the local roots (SYS0, for MOE; SYS1, for LARRY; and SYS2, for CURLY), and the common root (V4COMMON).

Each local root contains, in addition to the usual system directories, a SYSCOMMON directory that is an alias for the common root. The SYS$MANAGER:MAKEROOT.COM command procedure, which you run to create the local roots, sets up the alias by executing the following DCL command:

```
$ SET FILE/ENTER=DEV:[SYSn]SYSCOMMON.DIR DEV:[0,0]V4COMMON.DIR
```

2.4.2 Coordinating Shared System Files

To prepare a common user environment for a homogeneous cluster, you need to coordinate four system files:

- SYSUAF.DAT
- NETUAF.DAT
- VMSMAIL.DAT
- RIGHTSLIST.DAT

These files, which are part of the VAX/VMS operating system, contain information that controls such user functions as user logins, proxy login access, mail, and access to files. Together, they define the specific user environment for a single system. By coordinating these files from each node in a VAXcluster, you can define a specific user environment for the cluster.

Where you place the above four system files will depend upon whether your cluster is running from a *single* common system disk or some other system disk configuration. The other configurations are

- More than one common system disk
- Individual system disks
- A combination of one or more common system disks with one or more individual system disks

If your cluster is running from one common system disk, you should place the above four system files in SYSCOMMON:[SYSEXE]. If your cluster is running from any other system disk configuration, the four system files must be placed on a disk that is not MSCP served. Thus, you can use an HSC50-served disk or, in a two-node cluster, you can use a MASSBUS disk that is dual ported between both nodes.

In a totally heterogeneous cluster, each node uses its own version of each system file to maintain a unique user environment. For example, user logins to a particular cluster node are controlled by a SYSUAF.DAT file that is specific to that node.

If you upgraded to Version 4.0, carefully compare each file from each node and select the elements that should be common before actually building the common file. On new VAX/VMS systems, there is no need to compare the files because they are new. With new VAX/VMS systems, you should decide on the common elements and build the common file. The next three sections describe in detail the procedures for building a common version of each system file.

2.4.2.1 Coordinating User Accounts

In a homogeneous cluster, you must coordinate the user accounts from each node and build common versions of the following files:

- SYSUAF.DAT
- NETUAF.DAT

If your cluster consists of newly installed systems, you should build a common SYSUAF.DAT and NETUAF.DAT file as described in the *Guide to VAX/VMS System Management and Daily Operations*. Since the SYSUAF.DAT file on new VAX/VMS systems is empty except for the four DIGITAL-supplied accounts, very little coordination is necessary.

To build a common SYSUAF.DAT and NETUAF.DAT on a cluster composed of newly installed VAX/VMS systems, do the following:

1 Boot one of the cluster nodes in the single-system environment (see Section 2.2).

2 Log in to the SYSTEM account.

3 Run the Authorize Utility (AUTHORIZE) to add any accounts to the SYSUAF.DAT and NETUAF.DAT file on that node. (See the *VAX/VMS Utilities Reference Volume* for a description of AUTHORIZE.)

4 Use the new versions of SYSUAF.DAT and NETUAF.DAT as the common files.

If the nodes in your cluster were upgraded to Version 4.0, SYSUAF.DAT and NETUAF.DAT contain the records of established accounts. These accounts may be different from node to node. Therefore, if you are building a homogeneous cluster, you must coordinate and combine the SYSUAF.DAT and NETUAF.DAT files from each node to create common versions of these files.

For a more detailed discussion of the steps for building a common SYSUAF.DAT file on an upgraded system, see Appendix A.

The procedure for creating a common NETUAF.DAT file on upgraded systems is basically the same as the procedures for creating a common SYSUAF.DAT. The main difference is that less coordination is needed when merging the individual NETUAF.DAT files. For example, UICs are not used in the NETUAF records, and therefore no coordination is needed.

You should decide which existing proxy login records you want to keep on the cluster and include these records in the common NETUAF.DAT file. As with the SYSUAF.DAT files, you can use the Convert Utility to merge the NETUAF.DAT file from each node to create a common NETUAF.DAT file.

Once you have created common versions of SYSUAF.DAT and NETUAF.DAT you can set up each of them as either a common file on a shared disk or separate duplicate files.

If your cluster is running from one common system disk, all you need to do is to make sure that SYSUAF.DAT and NETUAF.DAT are included in SYSCOMMON:[SYSEXE]. No further action is required on your part.

If your cluster is running from any other system disk configuration, you must decide where to locate SYSUAF.DAT and NETUAF.DAT. Once you have placed these two files in a directory, you must define cluster-wide logical names to point to them.

Assume that the disk WORK5: is a volume shared by all nodes in the cluster and that it contains a common SYSUAF.DAT and NETUAF.DAT file. The commands in the following example define system logical names that point to the location of the common files:

```
$ DEFINE/SYSTEM/EXEC SYSUAF WORK5:[SYSEXE]SYSUAF
$ DEFINE/SYSTEM/EXEC NETUAF WORK5:[SYSEXE]NETUAF
```

You must add the DEFINE commands to the common SYSTARTUP command file. After you have copied the files to the appropriate directory in the shared disk volume, you should delete these files from the system disk in order to avoid possible confusion.

2.4.2.2 Preparing the MAIL Database

In a homogeneous environment, you may want to prepare a common mail database to allow users to use the Mail Utility (MAIL) to send and read their MAIL messages from any node in the cluster.

Each time MAIL executes in a single-system environment, it accesses a database file named SYS$SYSTEM:VMSMAIL.DAT. To set up VMSMAIL.DAT as a common file, define VMSMAIL to be the complete file specification of the common file by specifying the DEFINE command in the following format:

```
$ DEFINE/SYSTEM/EXEC VMSMAIL file-spec
```

You must make sure that you specify the DEFINE command before you invoke MAIL for the first time. When invoked for the first time, MAIL creates the database file, VMSMAIL.DAT, in SYS$SYSTEM by default. By defining VMSMAIL to be the location of a common file on a shared disk, you cause MAIL to create and use that file.

If your cluster is running from one common system disk, define VMSMAIL to be SYS$COMMON:[SYSEXE]VMSMAIL and invoke the Mail Utility, by issuing the following two commands:

```
$ DEFINE/SYSTEM/EXEC VMSMAIL SYSCOMMON:[SYSEXE]VMSMAIL
$ MAIL
```

The VMSMAIL file will be created in the common system directory. You will no longer need to use the logical name and you will not have to make any changes to the SYSTARTUP command file.

If your cluster is running from any other system disk configuration, you must decide where to locate the common VMSMAIL file. (Typically, you would place this file in the same directory in which SYSUAF and NETUAF reside.) You then define a logical name for the file and invoke the Mail Utility, by issuing the following commands:

```
$ DEFINE/SYSTEM/EXEC VMSMAIL WORK1:[SYSEXE]VMSMAIL
$ MAIL
```

You must also add the DEFINE command to the common SYSTARTUP command file.

The command in the above example defines VMSMAIL to be a file located in [SYSEXE] on the shared disk volume WORK1:. The first time MAIL is invoked, VMSMAIL.DAT is created in WORK1:[SYSEXE]. Subsequently, MAIL uses this file as the database.

2.4.2.3 Preparing the Rights Database

In a homogeneous cluster, you can create a common version of the rights database. Basically, the rights database is a file that associates users of the system or cluster with special names they are allowed to hold called identifiers. The rights database file, RIGHTSLIST.DAT, lies at the foundation of the ACL-based protection scheme. For more information on ACLs, see the description in the *Guide to VAX/VMS System Security*.

The cluster or security manager maintains the rights database, adding and removing identifiers as needs change. By allowing groups of users to hold identifiers, the manager has now created a different kind of group designation than the one used with the user's UIC. This alternative grouping is more finely tailored to the uses the holders of the identifier are expected to make of objects. It also permits each user to be a member of multiple overlapping groups.

For information on how the rights database is set up at the local node level, see the description in the Authorize Utility (AUTHORIZE) in the *VAX/VMS Utilities Reference Volume.*

If your cluster is running from one common system disk, the installation or upgrade procedure will place the RIGHTSLIST.DAT file in SYSCOMMON:[SYSEXE]. No further action is required on your part.

If your cluster is running from any other system disk configuration, copy SYS$SYSTEM:RIGHTSLIST.DAT to the directory in which you placed the SYSUAF, NETUAF, and VMSMAIL system files. Define a cluster-wide logical name for the RIGHTSLIST.DAT file. For example:

```
$ DEFINE/SYSTEM/EXEC RIGHTSLIST  WORK5:[SYSEXE]RIGHTSLIST
```

You must also add this DEFINE command to the common SYSTARTUP command file.

2.5 System Time on the Cluster

When a node joins the cluster, an attempt is made by the cluster to set the joining node's system time to the current time on the cluster. Although it is likely that the system time will be similar in each cluster node, there is no assurance that the time will be set. Also, no attempt is made to ensure that the system times remain similar throughout the cluster. (For example, there is no protection against different nodes having different clock rates.)

Note that the DCL command SET TIME and the system service $SETIME change the time *only* on the node on which they are executed.

3 Managing Cluster Queues

On a single-node system, print and batch job processing is limited to a single processor and local devices. On a VAXcluster, however, nodes can share device and processing resources. This ability to share resources allows for better workload balancing because batch and print job processing can be distributed across the cluster.

You control how jobs share device and processing resources on a VAXcluster by setting up and maintaining cluster queues. The strategy you use to set up and manage these queues will determine how well you match workloads to your cluster's device and processor resources.

You establish and control cluster queues with the same commands you use to manage queues on a single-node VAX/VMS system. These commands are described in the *VAX/VMS DCL Dictionary*. The sections that follow describe how to set up queues on a VAXcluster. The chapter assumes some knowledge of single-node queue management. Single-node queue management concepts are described in the *Guide to VAX/VMS System Management and Daily Operations*.

3.1 Cluster-wide Queues

Cluster-wide queues are controlled by a cluster-wide job controller queue file. This file makes queues available across the cluster and enables jobs to execute on any queue from any node, provided that the necessary mass storage volumes can be accessed by the node on which the job executes.

There can be only one job controller queue file on a cluster. If there is such a queue file, it must be on a disk that is accessible to the nodes participating in the cluster-wide queue scheme.

You control which nodes in the cluster share cluster-wide queues by specifying the location of the job controller queue file, JBCSYSQUE.DAT, with the DCL command START /QUEUE/MANAGER. You could use the following command string, for example, to set up a cluster-wide queue:

```
$ START/QUEUE/MANAGER SYS$COMMON:[SYSEXE]JBCSYSQUE.DAT
```

All nodes using queues must specify the same queue file in the START/QUEUE/MANAGER command.

3.2 Cluster Printer Queues

To establish printer queues, you should first decide on the type of queue configuration that will best suit your system. On a cluster, you have several alternatives that depend on the number and type of print devices you have on each node, and how you want print jobs to be processed. For example, make these decisions:

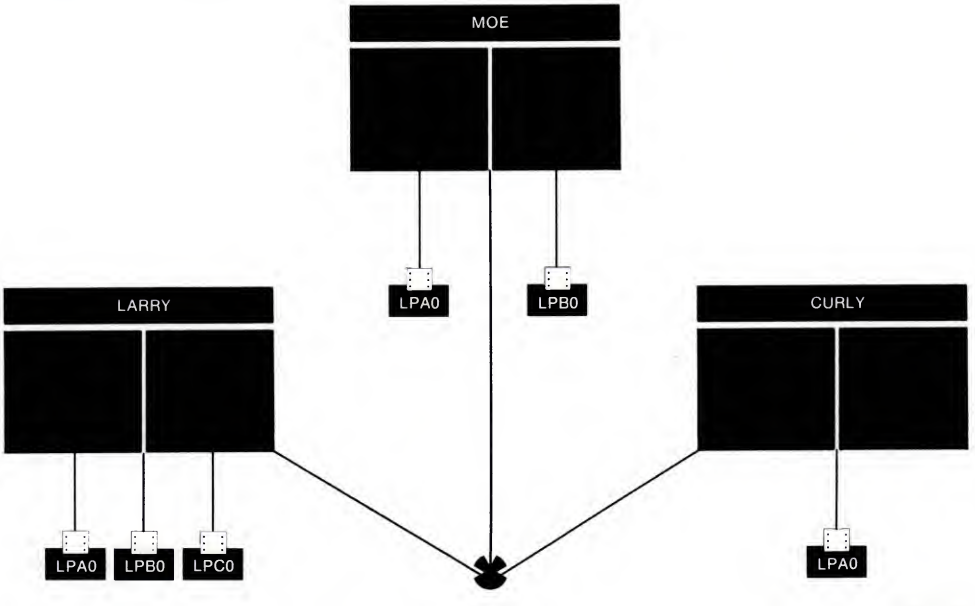
- Whether to set up generic printer queues that are local to each node
- Which printer queues should be assigned to any local generic queues
- Whether to set up any cluster-wide generic queues that will distribute print job processing across the cluster

Once you determine the strategy for your system, you can create a command procedure that will set up your queues. Figure 3–1 shows the printer configuration for a cluster consisting of the active nodes MOE, LARRY, and CURLY. The sections that follow will use this example configuration to illustrate various methods for establishing and naming printer queues on a VAXcluster. Sample command procedures are also included in Section 3.4 to serve as a guide to setting up queues.

3.2.1 Setting Up Printer Queues

You should set up printer queues using the same procedures that you would use for a single-node system (see the *Guide to VAX/VMS System Management and Daily Operations*). However, since each local node is part of the cluster system, you must provide a unique name for each queue you create in a cluster.

Figure 3–1 Sample Printer Configuration



You assign a unique name to a printer queue by specifying the DCL command INITIALIZE/QUEUE in the following format:

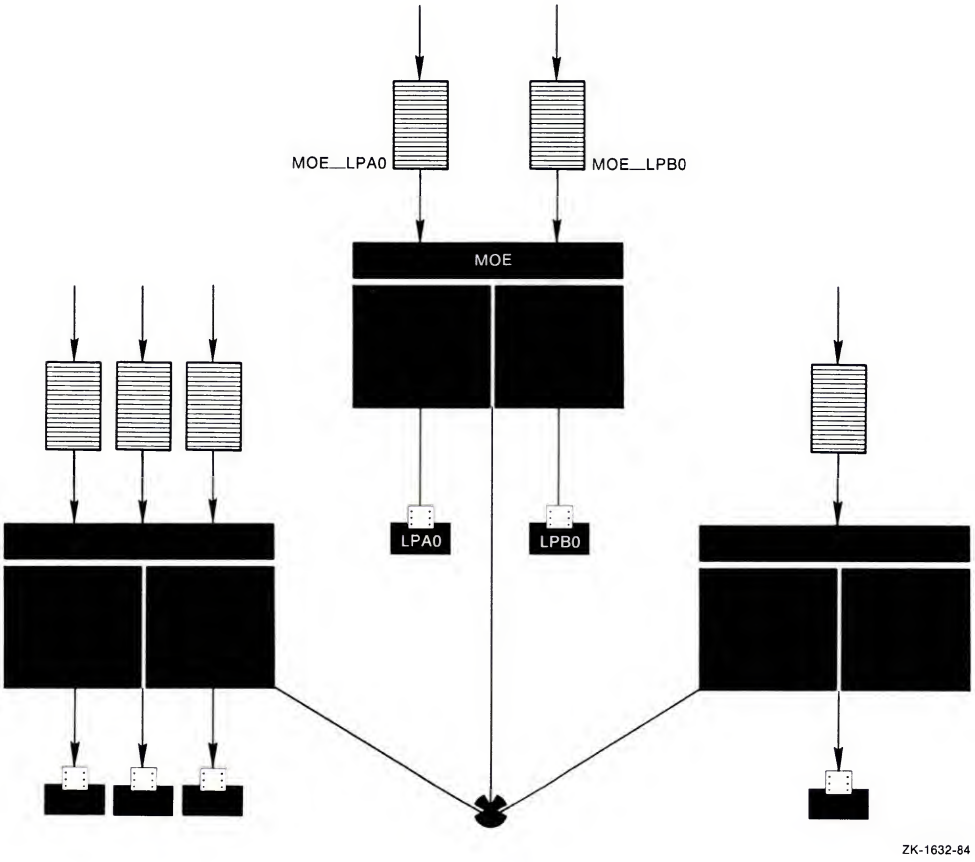
```
INITIALIZE/QUEUE/ON=node::device queue-name
```

The /ON qualifier specifies the node and printer that the queue is assigned to.

The commands in the following example make local printer queue assignments for the cluster node MOE shown in Figure 3–2:

```
$ INITIALIZE/QUEUE/ON=MOE::LPA0/START  MOE_LPA0
$ INITIALIZE/QUEUE/ON=MOE::LPB0/START  MOE_LPB0
```

Figure 3–2 Printer Queue Configuration

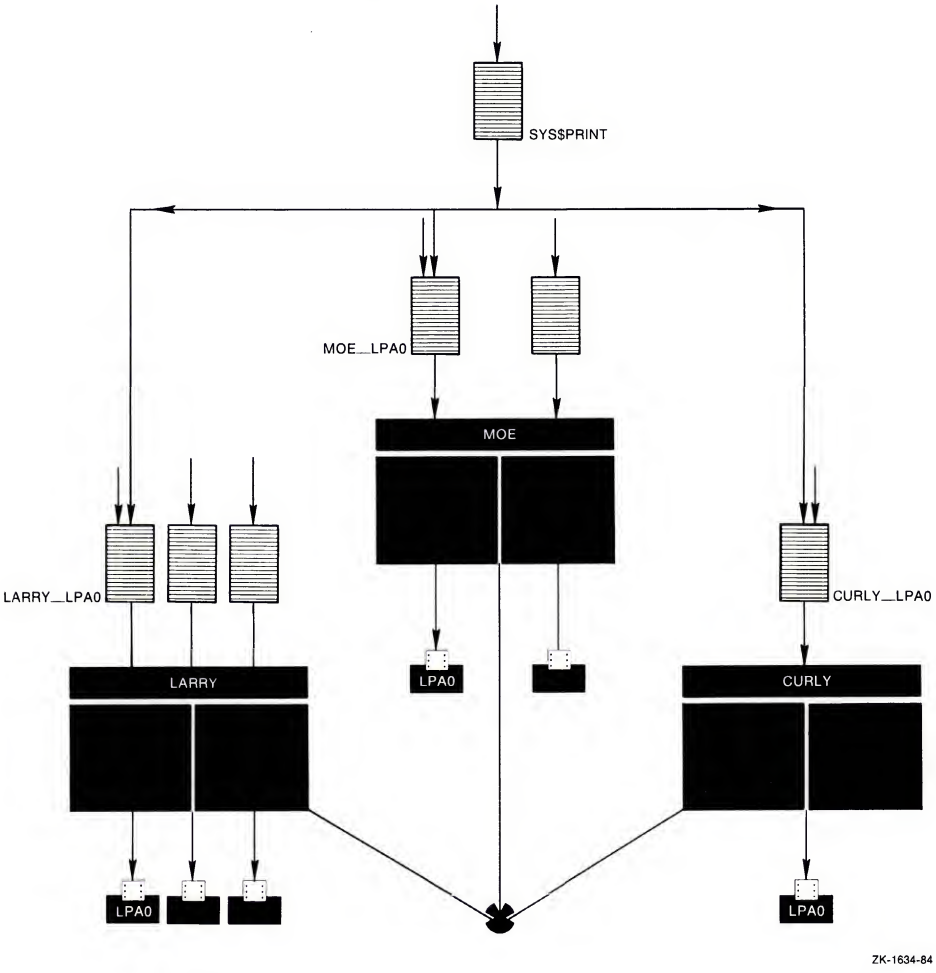


3.2.2 Setting Up Cluster-wide Generic Printer Queues

The cluster-wide job controller queue file enables you to establish generic queues that function throughout the cluster. Jobs queued to cluster-wide generic queues are placed in any assigned printer queue that is available, regardless of its location in the cluster. However, the file queued for printing must be accessible to the node to which the printer is connected.

Figure 3–3 illustrates a cluster-wide generic printer queue, in which the queues for all LPA0 printers in the cluster are assigned to a cluster-wide generic queue named SYS$PRINT.

Figure 3–3 Cluster Printer Queue Configuration With Cluster-wide Generic Printer Queue



The following command initializes and starts the cluster-wide generic queue SYS$PRINT:

```
$ INITIALIZE/QUEUE/GENERIC=(MOE_LPA0,LARRY_LPA0,-
  CURLY_LPA0)/START  SYS$PRINT
```

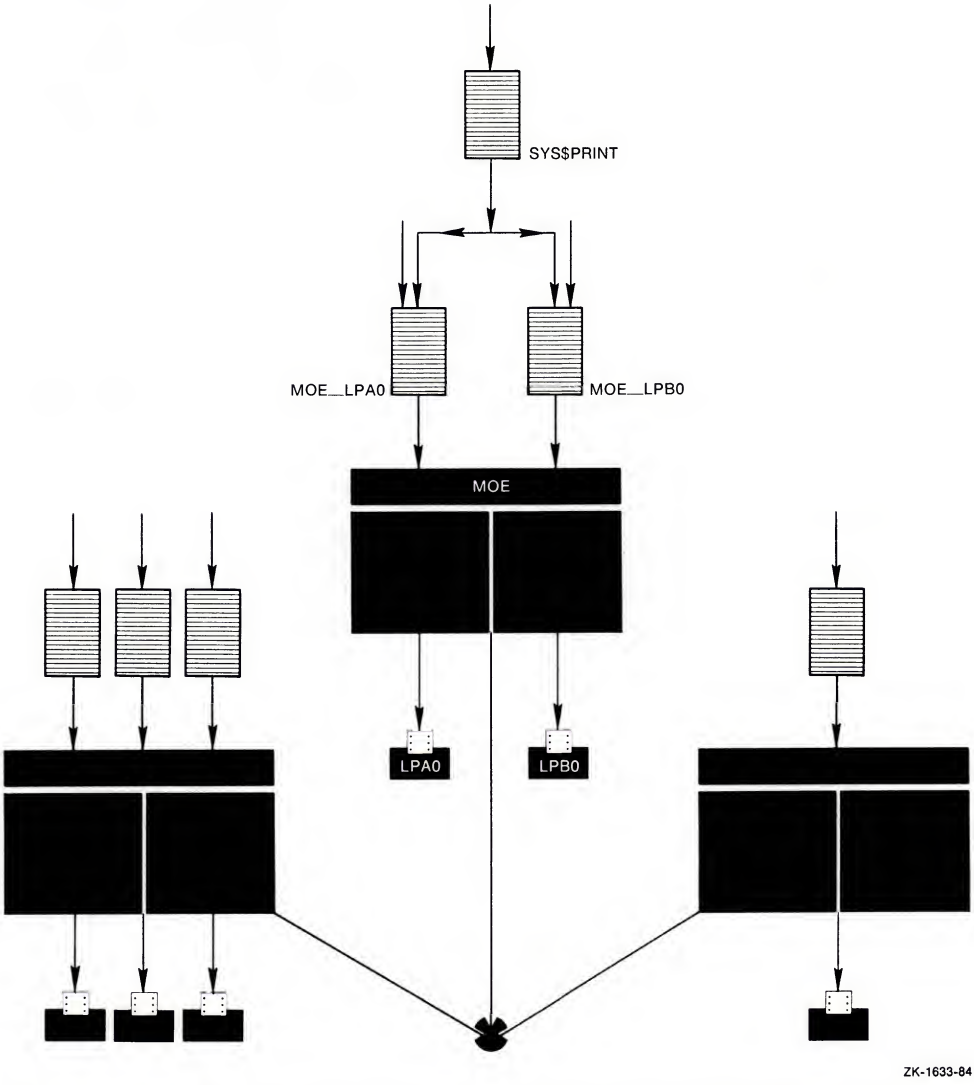
Jobs queued to SYS$PRINT are placed in whichever assigned printer queue is available. Thus, in this example, a print job from node MOE that is queued to SYS$PRINT may in fact be queued to MOE_LPA0, LARRY_LPA0, or CURLY_LPA0.

In addition to creating a queue for each local printer, you may wish to establish at least one local generic queue for similar devices on the local node. The following commands set up the local generic queue for node MOE shown in Figure 3–4.

```
$ INITIALIZE/QUEUE/GENERIC=(MOE_LPA0,MOE_LPB0)/START -
  MOE_PRINT
$ DEFINE/SYSTEM SYS$PRINT MOE_PRINT
```

In Figure 3–4 the generic printer queue MOE_PRINT is set up and explicitly assigned the printer queues MOE_LPA0 and MOE_LPB0.

Figure 3–4 Printer Queue Configuration With Local Generic Queue



In a single-node environment, you would name the generic queue SYS$PRINT, since print jobs are queued to SYS$PRINT by default. In a cluster, however, the separate nodes cannot have independent queues with the same name; therefore, you cannot create multiple generic queues named SYS$PRINT. To get around this problem, you can create a generic queue, assign it a unique queue name, and then establish a system-wide logical name equating SYS$PRINT to the generic queue name. This logical name assignment is system-wide on the local node, affecting operations on that node. Thus, only print jobs from users on MOE are queued to MOE_PRINT by default.

Since print jobs on each cluster node are queued to SYS$PRINT by default, you might want to establish SYS$PRINT as a cluster-wide generic printer queue that distributes print job processing throughout the cluster.

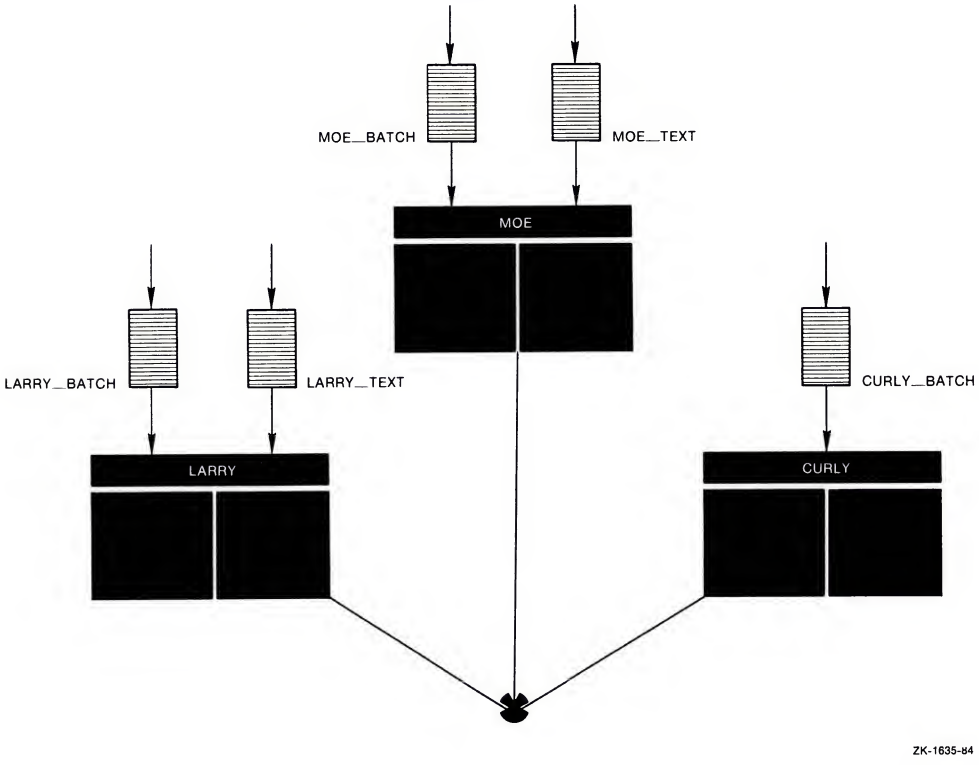
3.3 Cluster Batch Queues

Before you establish batch queues, you should first decide on the type of queue configuration that will best suit your cluster. As cluster manager it is up to you to set up batch queues to maintain efficient batch job processing on the cluster. For example, you should do the following:

- Determine what type of processing will be performed on each node
- Set up local batch queues that conform to these processing need
- Decide whether to set up any cluster-wide generic queues that will distribute batch job processing across the cluster

Once you determine the strategy that best suits your system needs, you can create a command procedure that will set up your queues. Figure 3–5 shows the batch queue configuration for a cluster consisting of the active nodes MOE, LARRY, and CURLY. The sections that follow will use this example configuration to illustrate various methods for establishing and naming batch queues on a VAXcluster. Sample command procedures for this configuration are also included in Section 3.4 to serve as a guide to setting up queues.

Figure 3–5 Sample Batch Queue Configuration



3.3.1 Setting Up Executor Batch Queues

Generally, you set up executor batch queues on each cluster node using the same procedures you use for a single-node system. For more detailed information on how this is done, see the *Guide to VAX/VMS System Management and Daily Operations.*

You assign a unique name to a batch queue by specifying the DCL command INITIALIZE/QUEUE in the following format:

```
INITIALIZE/QUEUE/ON=node:: queue-name
```

The /ON qualifier specifies the node on which the batch queue runs.

The commands in the following example make local batch queue assignments for the cluster node MOE shown in Figure 3–5:

```
$ INITIALIZE/QUEUE/BATCH/ON=MOE::/START  MOE_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=MOE::/START  MOE_TEXT
```

In a single-node environment, you would name one batch queue SYS$BATCH, since batch jobs are queued to SYS$BATCH by default. You may decide to follow this convention for each node in the cluster. In a cluster, however, the separate nodes cannot have independent queues with the same name; therefore you cannot create a queue named SYS$BATCH for each node in the cluster. To get around this problem, you can create a queue, assign it a unique queue name, and then establish a system-wide logical name equating SYS$BATCH to the queue name as follows:

```
$ INITIALIZE/QUEUE/BATCH/ON=MOE::/START MOE_BATCH
$ DEFINE/SYSTEM SYS$BATCH MOE_BATCH
```

This logical name definition is system-wide on the local node, affecting only operations on that node. Thus, only batch jobs from users on MOE are queued to MOE_BATCH by default.

Since batch jobs on each cluster node are queued to SYS$BATCH by default, you should consider establishing SYS$BATCH as a cluster-wide generic batch queue that distributes batch job processing throughout the cluster. Note, however, that you should do this only if your cluster is fairly homogeneous. Guidelines for establishing cluster-wide generic batch queues are presented in the following section.

3.3.2 Setting Up Generic Batch Queues

Unlike a printer queue, a batch queue can be set up to allow more than one job to execute simultaneously. For this reason it is often not necessary on a single-node system to create more than one batch queue of the same type and assign them to a generic batch queue.

On a cluster, however, where you have multiple processors, you may want to distribute batch processing across the nodes to balance the use of processing resources. You can achieve this workload distribution by assigning local batch queues to one or more cluster-wide generic batch queues. These generic batch queues control batch processing over the cluster by placing batch jobs in assigned batch queues that are available.

Instead of having a queue named SYS$BATCH set up on each cluster node (as described in Section 3.3.1), you can create a cluster-wide generic batch queue and name it SYS$BATCH.

For example, in Figure 3–6 batch queues from each node are assigned to a cluster-wide generic batch queue named SYS$BATCH. Users can submit a job to a specific queue, or if they have no special preference, submit it by default to the cluster-wide generic queue, SYS$BATCH. The generic queue in turn places the job in an available assigned queue in the cluster.

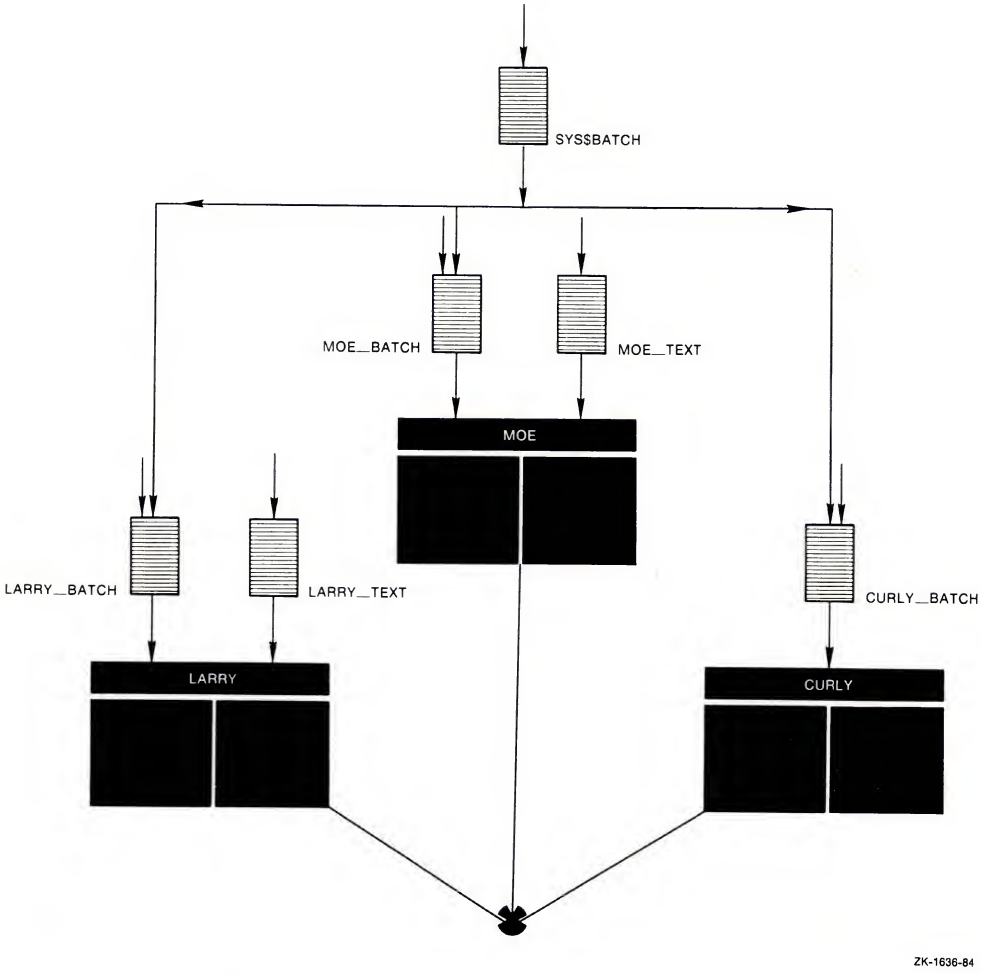
If more than one assigned queue is available, the system selects the queue that will minimize the ratio (executing jobs/job limit) for all assigned queues.

3.4 Command Procedures for Establishing Queues

To configure queues on a cluster properly, you must coordinate, between the nodes in the cluster, the commands in the procedures that initialize and start queues. Each active node in a cluster must initialize its local queues as well as the queues of other cluster nodes, so that when new nodes join the cluster, queues are recognized by all the nodes. However, since the nodes in a cluster boot separately rather than simultaneously, a booting node must start only its own local queues.

As a rule, the startup command procedure for each active node in a cluster must initialize every queue in the cluster but start only its local queues and any cluster-wide generic queues.

Figure 3–6 Batch Queue Configuration With Cluster-wide Generic Queue



You should include commands to establish queues in the SYSTARTUP.COM procedure or in a separate command procedure file named, for example, STARTQ.COM that is invoked by SYSTARTUP.COM. DIGITAL suggests that you set up your STARTQ command procedure(s) as a common file on a

shared disk. In this case, the common STARTQ.COM file may reside on the same disk as the job-controller queue file.

3.4.1 Starting Queues Using Node-Specific Command Procedures

For each node in the cluster, either add node-specific queue commands to the node-specific SYSTARTUP.COM procedure or create a STARTQ command procedure that is invoked by the node-specific SYSTARTUP.COM procedure. Example 3–1 illustrates the use of separate node-specific command procedures to initialize and start the printer configuration shown in Figure 3–1 and the batch configuration shown in Figure 3–5.

Example 3–1 Setting Up Queues Using Separate Node-Specific Procedures

STARTQ Command Procedure for Node MOE

```
$ SET NOON
$ !
$ !  Start job queue manager.
$ !
$ START/QUEUE/MANAGER  WORK1:[CLUSMAN]
$ !
$ !  Initialize and start local printer queues.
$ !
$ INITIALIZE/QUEUE/ON=MOE::LPA0/START  MOE_LPA0
$ INITIALIZE/QUEUE/ON=MOE::LPB0/START  MOE_LPB0
$ !
$ !  Initialize remote printer queues.
$ !
$ INITIALIZE/QUEUE/ON=LARRY::LPA0  LARRY_LPA0
$ INITIALIZE/QUEUE/ON=LARRY::LPB0  LARRY_LPB0
$ INITIALIZE/QUEUE/ON=LARRY::LPC0  LARRY_LPC0
$ INITIALIZE/QUEUE/ON=CURLY::CURLY_LPA0  CURLY_LPA0
$ !
$ ! Initialize and start cluster-wide generic printer queue.
$ !
$ INITIALIZE/QUEUE/GENERIC=(MOE_LPA0,LARRY_LPA0,CURLY_LPA0)-
  /START SYS$PRINT
$ !
$ !  Initialize batch queues on local node.
$ !
$ INITIALIZE/QUEUE/BATCH/ON=MOE::/START  MOE_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=MOE::/START  MOE_TEXT
$ !
$ !  Initialize queues from other nodes.
$ !
$ INITIALIZE/QUEUE/BATCH/ON=LARRY::  LARRY_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=LARRY::  LARRY_TEXT
$ INITIALIZE/QUEUE/BATCH/ON=CURLY::  CURLY_BATCH
$ !
$ !  Initialize cluster-wide generic batch queue.
$ !
$ INITIALIZE/QUEUE/BATCH/GENERIC=(MOE_BATCH,LARRY_BATCH,-
  CURLY_BATCH)/START  SYS$BATCH
```

(Continued on next page)

Example 3–1 (Cont.) Setting Up Queues Using Separate Node-Specific Procedures

STARTQ Command Procedure for Node LARRY

```
$ SET NOON
$ !
$ !  Start job queue manager.
$ !
$ START/QUEUE/MANAGER  WORK1:[CLUSMAN]
$ !
$ !  Initialize and start local printer queues.
$ !
$ INITIALIZE/QUEUE/ON=LARRY::LPA0/START  LARRY_LPA0
$ INITIALIZE/QUEUE/ON=LARRY::LPB0/START  LARRY_LPB0
$ INITIALIZE/QUEUE/ON=LARRY::LPC0/START  LARRY_LPC0
$ !
$ !  Initialize remaining printer queues.
$ !
$ INITIALIZE/QUEUE/ON=MOE::LPA0  MOE_LPA0
$ INITIALIZE/QUEUE/ON=MOE::LPB0  MOE_LPB0
$ INITIALIZE/QUEUE/ON=CURLY::CURLY_LPA0  CURLY_LPA0
$ !
$ !  Initialize and start cluster-wide generic printer queue.
$ !
$ INITIALIZE/QUEUE/GENERIC=(MOE_LPA0,LARRY_LPA0,-
  CURLY_LPA0)/START SYS$PRINT
$ !
$ !  Initialize batch queues on local node.
$ !
$ INITIALIZE/QUEUE/BATCH/ON=LARRY::/START  LARRY_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=LARRY::/START  LARRY_TEXT
$ !
$ !  Initialize queues from other nodes.
$ !
$ INITIALIZE/QUEUE/BATCH/ON=MOE::  MOE_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=MOE::  MOE_TEXT
$ INITIALIZE/QUEUE/BATCH/ON=CURLY::  CURLY_BATCH
$ !
$ !  Initialize cluster-wide generic batch queue.
$ !
$ INITIALIZE/QUEUE/BATCH/GENERIC=(MOE_BATCH,LARRY_BATCH,-
  CURLY_BATCH) SYS$BATCH
```

(Continued on next page)

Example 3–1 (Cont.) Setting Up Queues Using Separate Node-Specific Procedures

STARTQ Command Procedure for Node CURLY

```
$ SET NOON
$ !
$ !  Start job queue manager.
$ !
$ START/QUEUE/MANAGER  WORK1:[CLUSMAN]
$ !
$ !  Initialize and start local printer queue.
$ !
$ INITIALIZE/QUEUE/ON=CURLY::LPA0/START  CURLY_PRINT
$ !
$ !  Initialize remaining printer queues.
$ !
$ INITIALIZE/QUEUE/ON=MOE::LPA0  MOE_LPA0
$ INITIALIZE/QUEUE/ON=MOE::LPB0  MOE_LPB0
$ INITIALIZE/QUEUE/ON=LARRY::LPA0  LARRY_LPA0
$ INITIALIZE/QUEUE/ON=LARRY::LPB0  LARRY_LPB0
$ INITIALIZE/QUEUE/ON=LARRY::LPC0  LARRY_LPC0
$ !
$ !  Initialize and start cluster-wide generic printer queue.
$ !
$ INITIALIZE/QUEUE/GENERIC=(MOE_LPA0,LARRY_LPA0,-
  CURLY_LPA0)/START SYS$PRINT
$ !
$ ! Initialize batch queues on local node.
$ !
$ INITIALIZE/QUEUE/BATCH/ON=CURLY::/START  CURLY_BATCH
$ !
$ !  Initialize queues from other nodes.
$ !
$ INITIALIZE/QUEUE/BATCH/ON=MOE::  MOE_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=MOE::  MOE_TEXT
$ INITIALIZE/QUEUE/BATCH/ON=LARRY::  LARRY_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=LARRY::  LARRY_TEXT
$ !
$ !  Initialize cluster-wide generic batch queue.
$ !
$ INITIALIZE/QUEUE/BATCH/GENERIC=(MOE_BATCH,LARRY_BATCH,-
  CURLY_BATCH) SYS$BATCH
```

In Example 3–1, each command procedure performs the following operations for the specific node:

- Starts the system job queue manager
- Specifies the location of the job-controller queue file

- Initializes and starts each local queue on the local node
- Initializes all other queues from other nodes
- Initializes and starts the cluster-wide generic printer queue SYS$PRINT
- Initializes and starts the cluster-wide generic batch queue SYS$BATCH

3.4.2 Starting Queues Using a Common Command Procedure

You can create a common command procedure, named for example, STARTQ.COM, and store it on a shared disk. Using this method, each node can share the same copy of the common STARTQ.COM procedure. Each node invokes the common STARTQ.COM procedure from the common version of SYSTARTUP.COM. You can also include the commands to set up queues in the common SYSTARTUP.COM file instead of a separate STARTQ.COM file.

Example 3–2 illustrates the use of a common STARTQ command procedure on a shared disk to initialize and start the printer queues shown in Figure 3–1.

Example 3–2 Starting Queues Using a Common Command Procedure

```
$ !
$ !  Compute the name of the executing node.
$ !
$ NODE = F$GETSYI("NODENAME")
$ !
$ MOE_START = "/NOSTART"
$ LARRY_START = "/NOSTART"
$ CURLY_START = "/NOSTART"
$ !
$ ! Redefine one of the previous symbols.
$ !
$ 'NODE'_START = "/START"
$ !
$ !
```

(Continued on next page)

Example 3–2 (Cont.) Starting Queues Using a Common Command Procedure

```
$ SET NOON
$ !
$ ! Start up the job controller.
$ !
$ START/QUEUE/MANAGER  WORK1:[CLUSMAN]
$ !
$ !     Set up printer queues.
$ !     Initialize all nodes. Start local node only.
$ !
$ INITIALIZE/QUEUE/ON=MOE::LPA0 'MOE_START'  MOE_LPA0
$ INITIALIZE/QUEUE/ON=MOE::LPB0 'MOE_START'  MOE_LPB0
$ !
$ INITIALIZE/QUEUE/ON=LARRY::LPA0 'LARRY_START'  LARRY_LPA0
$ INITIALIZE/QUEUE/ON=LARRY::LPB0 'LARRY_START'  LARRY_LPB0
$ INITIALIZE/QUEUE/ON=LARRY::LPC0 'LARRY_START'  LARRY_LPC0
$ !
$ INITIALIZE/QUEUE/ON=CURLY::LPA0 'CURLY_START' -
  CURLY_PRINT
$ !
$ ! Set up main batch queues.
$ !
$ INITIALIZE/QUEUE/BATCH/ON=MOE::/JOB=6/WSEXTENT=500 -
  'MOE_START' MOE_BATCH
$ !
$ INITIALIZE/QUEUE/BATCH/ON=LARRY::/JOB=5/WSEXTENT=600 -
  'LARRY_START' LARRY_BATCH
$ !
$ INITIALIZE/QUEUE/BATCH/ON=CURLY/JOB=6/WSEXTENT=600 -
  'CURLY_START' CURLY_BATCH
$ !
$ ! Set up batch processing queues.
$ !
$ INITIALIZE/QUEUE/BATCH/ON=MOE::/JOB=2/WSEXTENT=1500 -
  'MOE_START'  MOE_TEXT
$ !
$ INITIALIZE/QUEUE/BATCH/ON=LARRY::/JOB=2/WSEXTENT=1500 -
  'LARRY_START' LARRY_TEXT
$ !
$ ! Set up cluster-wide generic batch processing queue.
$ !
$ INITIALIZE/QUEUE/BATCH/GENERIC=(MOE_BATCH,LARRY_BATCH,-
  CURLY_BATCH) SYS$BATCH
```

The command procedure in Example 3–2 performs the same queue setup operations as the command procedures shown in Example 3–1. The STARTQ file in this example, however, executes a common set of commands that function according to the node executing them. A set of conditional symbols are

assigned to control whether queues are started. In this way, each node initializes all the queues in the cluster but starts only its own.

3.5 Summary of Commands for Setting Up Cluster Queues

The following is a summary of commands used to set up queues on a VAXcluster.

- (Start the system job queue manager)

```
$ START/QUEUE/MANAGER file-spec
```

- (Set up printer queues)

```
$ INITIALIZE/QUEUE/ON=node::device queue-name
$ INITIALIZE/QUEUE/ON=node::device/START -
  queue-name
```

- (Set up generic printer queues)

```
$ INITIALIZE/QUEUE/GENERIC=(queue1,queue2...)-
  /START queue-name
```

- (Set up batch queues)

```
$ INITIALIZE/QUEUE/BATCH/ON=node:: queue-name
$ INITIALIZE/QUEUE/BATCH/ON=node::/START -
  queue-name
```

- (Set up generic batch queues)

```
$ INITIALIZE/QUEUE/BATCH/GENERIC=(queue1,-
  queue2...)/START queue-name
```

4 Managing Cluster Disks

On a VAXcluster, there are two basic types of disk and tape devices:

- Restricted-access devices, which are accessible only by the local node or nodes to which they are connected
- Cluster-accessible devices, which are accessible by any node in the cluster.

A disk or magnetic tape device connected to an HSC50 is by design a cluster-accessible device. However, any other disk device, such as a MASSBUS, UNIBUS, or UDA disk, is a restricted-access device unless you explicitly set it up as a cluster-accessible device.

As cluster manager, you are responsible for planning, organizing, and setting up the proper cluster device configuration for your site. You should decide which disk devices should have access restricted to the local node, and which should be accessible to the cluster. For example, you may want to restrict access of a particular disk to the users on the node connecting the device. Or, you may decide to set up a disk as a cluster-accessible device, so that any user on any cluster node can allocate and use it.

Once you have planned your configuration strategy, you can use the steps outlined in this chapter to set up cluster devices. This chapter describes how to set up, name, and manage disk devices in a VAXcluster. It also describes how to manage shared disk volumes. Sample command procedures are provided to serve as guides for setting up cluster disks and mounting shared disk volumes.

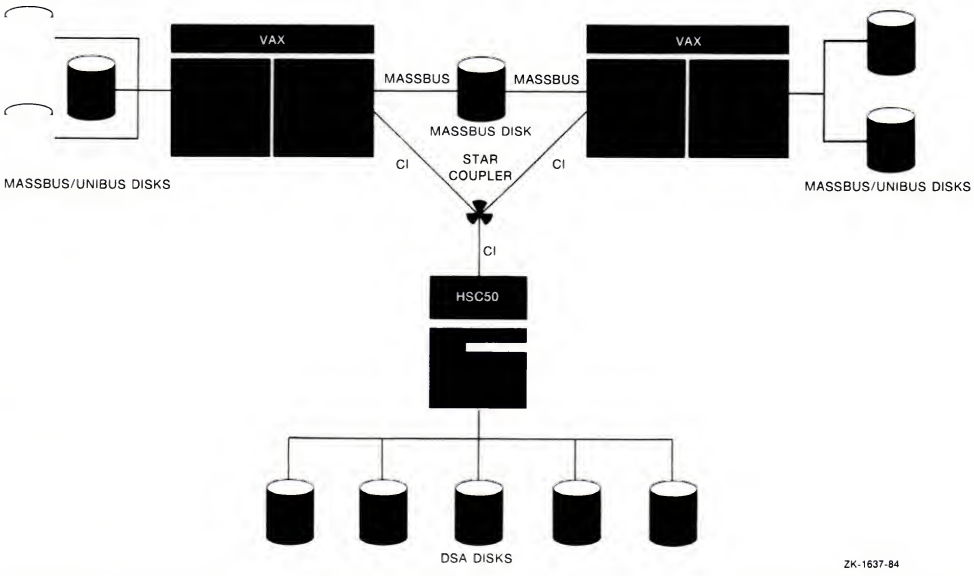
4.1 Cluster-Accessible Disks

A *cluster-accessible disk* is a disk that every node in the cluster can recognize and access. The following types of disks are cluster accessible:

- HSC disks
- MSCP served disks
- Dual-ported MASSBUS disks
- Dual-pathed disks

Figure 4–1 illustrates the types of cluster-accessible disks used on VAXclusters.

Figure 4–1 Disk Configuration Within Shared Disks



4.1.1 HSC Disks

An HSC disk is a DIGITAL Storage Architecture (DSA) disk that is connected to an HSC50.

If an HSC50 is a member of the cluster, devices connected to the HSC50 are automatically accessible by any node in the cluster. You can also set up HSC disks to be dual pathed between two HSC50s. Dual-pathed disks are described in Section 4.1.4.

4.1.2 MSCP-Served Disks

MSCP is the protocol used to communicate between a VAX host and a DSA controller. The MSCP server enables a VAX processor to make locally connected MASSBUS and UNIBUS disks available to all other users of the VAXcluster.

Unlike HSC devices, MASSBUS, UNIBUS, and UDA disks that are connected to a VAX node in the cluster are not automatically cluster accessible. Access to these devices is restricted to the local node unless you explicitly set them up as cluster accessible.

To make a non-HSC disk accessible to all cluster nodes, the MSCP server must be loaded on the local node and it must be instructed to make the disk available cluster wide.

The SYSGEN command MSCP loads and starts the MSCP server, and the DCL command SET DEVICE/SERVED is used to make a particular disk available to all of the nodes in the cluster.

Typically, you make local disks available to the cluster by including SET DEVICE/SERVED commands in the appropriate startup command procedure for each node. You can also include these commands in a separate procedure that is invoked by the startup command procedure (see Section 4.4). Except for the system disk, which is mounted automatically, all non-HSC disks must be served before they are mounted.

Note: Do not add a disk to the MSCP server database that is not local to the node adding the disk.

The following example shows the section of a command procedure that loads the MSCP server and serves two disks:

```
$ RUN SYS$SYSTEM:SYSGEN
  MSCP
  EXIT
$ SET DEVICE/SERVED MOE$DBA1
$ SET DEVICE/SERVED MOE$DBA2
```

You can also use the SET DEVICE/SERVED command at the DCL level to accomplish this task interactively. For example, the following command string

```
$ SET DEVICE/SERVED LARRY$DRA4:
```

makes the local disk LARRY$DRA4: available to the members of the cluster. Note that in this case the local node must be a member of a VAXcluster and the local MSCP server must have been invoked by SYSGEN.

Note that unless the disk device that you intend to make available to the cluster is a system disk, it must not already be mounted when you issue the SET DEVICE/SERVED command. For more information on the DCL command SET DEVICE /SERVED, see the description in the *VAX/VMS DCL Dictionary*.

Locally connected disks that are not served cannot be accessed by other members of the cluster. They remain accessible only to the local node.

4.1.3 Dual-Ported MASSBUS Disks

A dual-ported MASSBUS disk can be accessed by the nodes connecting it. If the cluster is comprised of only the two VAX nodes that are connecting the disk, the disk is considered cluster accessible.

Note: The disk must have the same device name on both nodes connecting it.

Dual-ported MASSBUS disks on a VAXcluster are an extension of the support offered in VAX/VMS Version 3.0. On a VAXcluster, when two active members have MASSBUS disks dual ported between them and the ALLOCLASS system parameters (see Section 4.2.2) on both nodes are identical, Files–11 ODS-2 volumes can be mounted READ/WRITE in the

dual-ported drives by both nodes. The VAXcluster distributed files system synchronizes access to the file structure on such disks.

To set up a dual-ported MASSBUS disk, specify the DCL command **SET DEVICE** in the following format before mounting the disk:

```
$ SET DEVICE/DUAL_PORTED device-name
```

Note: A MASSBUS disk may be used as either a dual-ported disk or a system disk, but not both.

On clusters with more than two active nodes, you can set up a dual-ported MASSBUS disk to be cluster accessible by making the dual-ported disk *dual pathed*. Setting up a disk as dual pathed makes the disk accessible to all cluster nodes, not just the nodes connecting the disk. Dual-pathed disks are described in Section 4.1.4.

4.1.4 Dual-Pathed Disks

A *dual-pathed disk* is a dual-ported disk that is accessible to all the nodes in the cluster, not just the nodes connecting the disk. A dual-pathed disk can be either of the following:

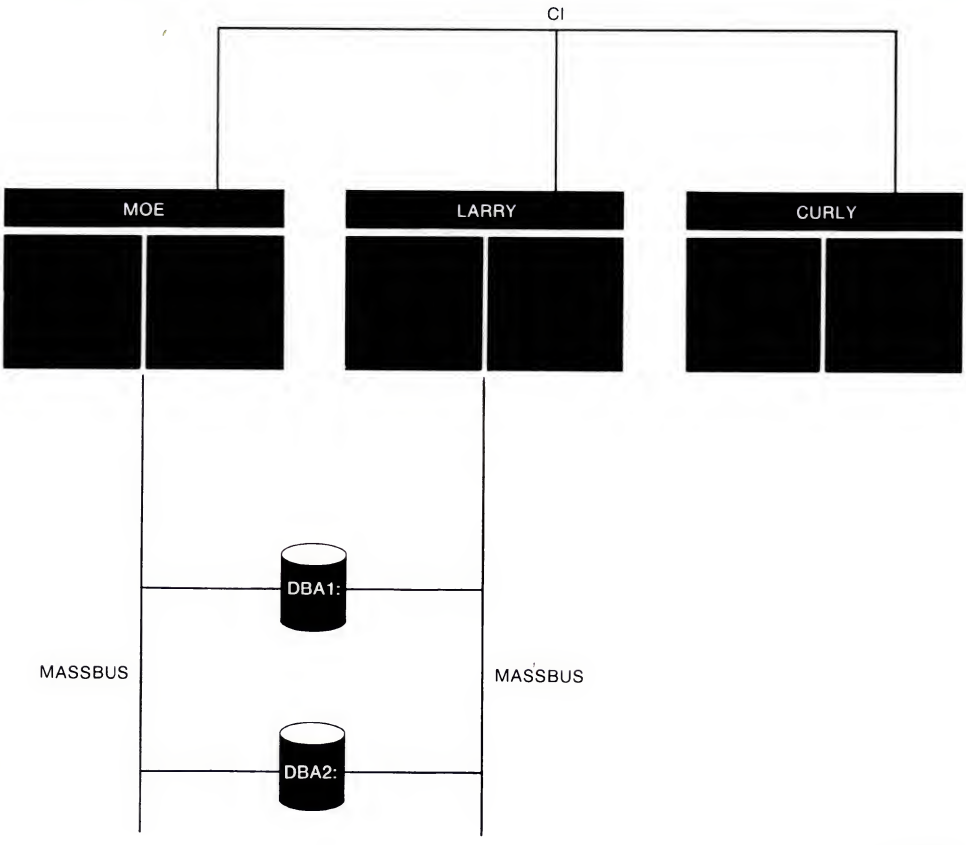
- A dual-ported MASSBUS disk that is cluster accessible by way of the MSCP server
- A dual-ported HSC disk

The term *dual-pathed* refers to the two paths through which cluster nodes can access a disk that they are not directly connected to. If one path fails, the disk is accessed over the other path. (Note that with a dual-pathed MASSBUS disk, a node connected to the disk always accesses it locally.)

Figure 4–2 shows a VAXcluster with a dual-pathed disk dual ported between nodes MOE and LARRY. Assuming that the disk is accessible to the cluster by way of the MSCP server, the node CURLY can access the disk through either of two paths—the path through MOE, or the path through LARRY.

You must create a unique name for the dual-pathed disk by assigning an allocation class to the nodes (VAX or HSC50) connecting the disk. Section 4.2.2 describes allocation classes.

Figure 4–2 Cluster Configuration Within Dual-Pathed MASSBUS Disks

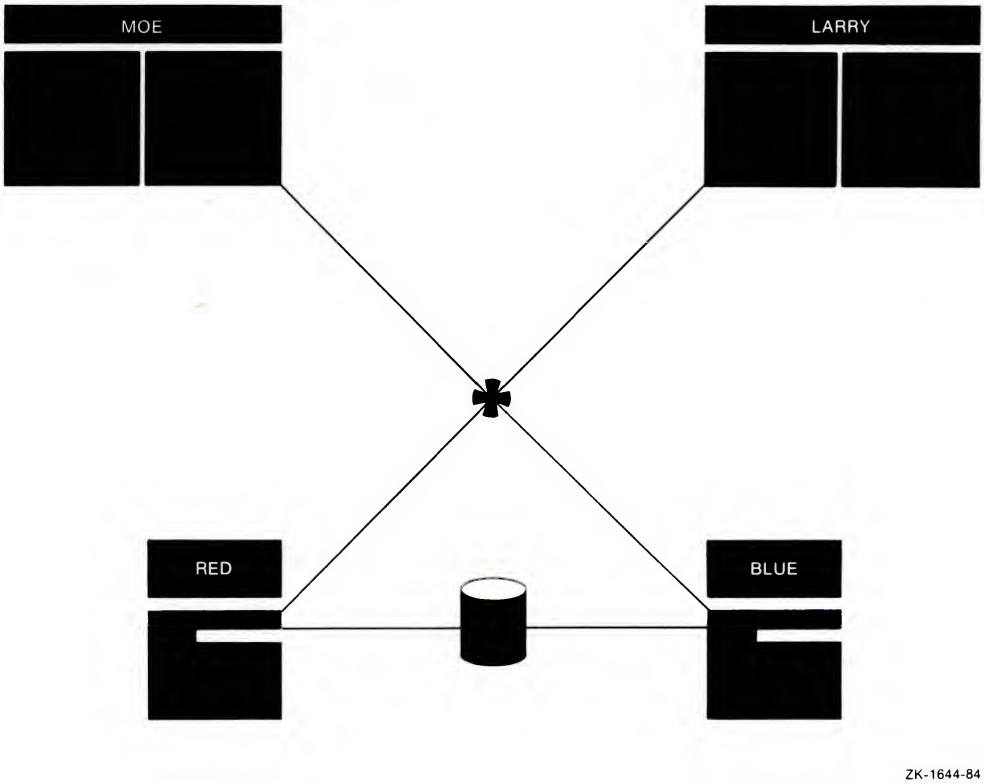


4.1.4.1 Dual-Pathed HSC Disks

By design, HSC disks are cluster accessible. Therefore, if they are dual ported, they are automatically dual pathed. Cluster nodes can access a dual-pathed HSC disk by way of a path through either HSC connecting the device.

Figure 4–3 shows a cluster configuration with a dual-ported HSC disk. Because the HSC devices are dual pathed, nodes MOE and LARRY can access them using the path through either RED or BLUE.

Figure 4–3 Cluster Configuration With a Dual-Ported HSC Disk



For each dual-ported HSC disk, you can control failover to a specific port using the port select buttons on the front of each drive. By pressing either port select button (A or B) on a particular drive, you can cause the device to failover to the specified port.

With the port select buttons, you can select alternate ports to balance the disk controller workload between two HSC50s. For example, you could set half of your disks to use Port A and set the other half to use Port B.

The port select buttons also enable you to failover all the disks to an alternate port manually when you anticipate the shutdown of one of the HSC50s.

4.1.4.2 Dual-Pathed MASSBUS Disks

To set up a dual-ported MASSBUS disk to be dual pathed, use the SET DEVICE/SERVED command (as described in Section 4.1.2) on both VAX nodes connecting the disk.

4.2 Cluster Device-Naming Conventions

To manage cluster disks properly, it is important to understand the conventions used to identify them. Every device in a VAXcluster is identified by a unique name, which provides a way to accurately access devices in the cluster. The next two sections describe the conventions used in naming cluster disk and tape devices.

4.2.1 Cluster Device Names

Each device in a VAXcluster is identified by a unique name. Except for dual-pathed disks, cluster devices are identified by a cluster device name specification in the format

```
node$device-name
```

Node is the name of the node (VAX or HSC50) to which the device is attached, and **device-name** is the physical device name. Figure 4–4 illustrates the use of cluster device names on a VAXcluster.

Devices that are local to a cluster node can be accessed by the local node through the traditional device name (for example DBA1:) or the cluster device name.

4.2.2 Allocation Class Identifiers

A disk that is dual pathed between two nodes is identified using a naming convention that is different from that discussed in Section 4.2.1.

Each time a node that is not connecting a dual-pathed disk tries to access the disk, the choice of which path to take is made arbitrarily, since no path to the disk is ever guaranteed. Because the access path to a dual-pathed disk is chosen without regard to the names of the nodes (VAX or HSC50) serving the device, a unique path-name independent identifier is required for the disk.

Figure 4–4 Cluster Device Names

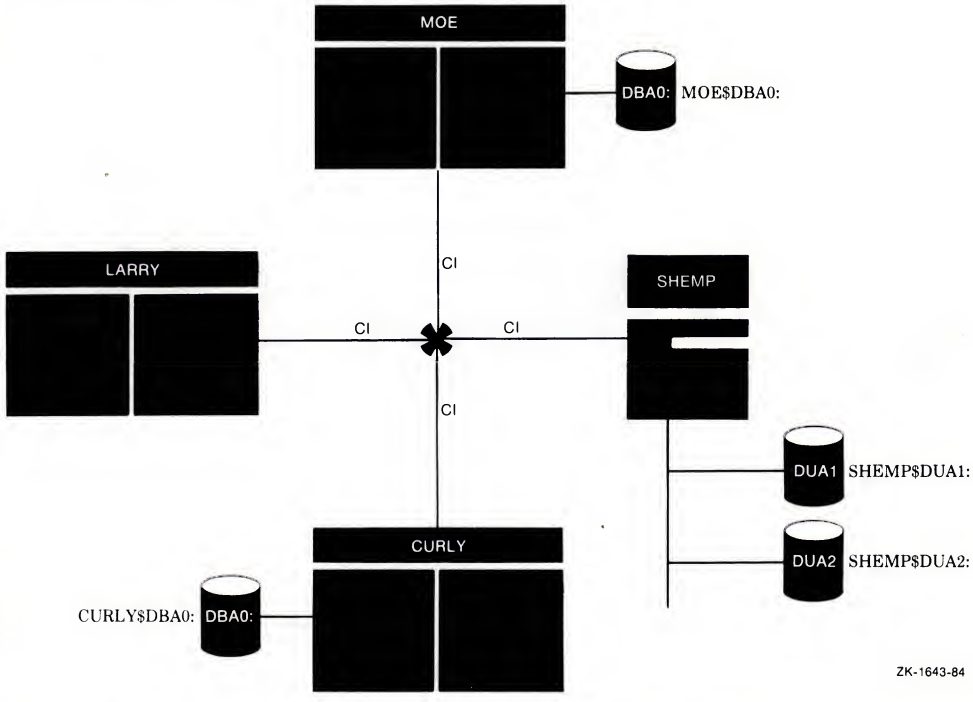
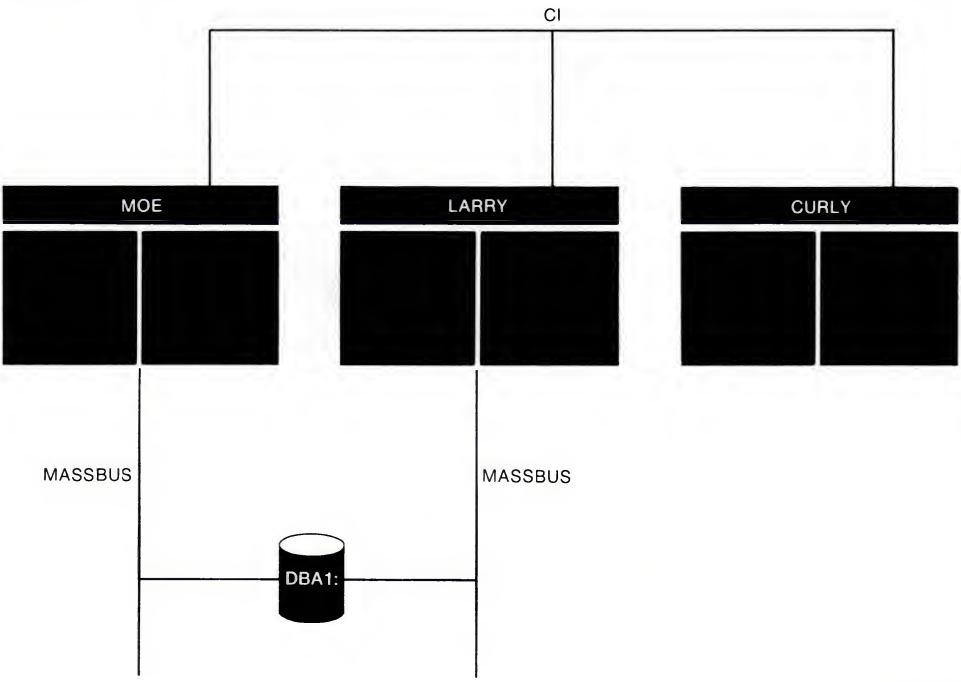


Figure 4–5 shows a three-node cluster with a MASSBUS disk dual pathed between nodes MOE and LARRY. Nodes MOE and LARRY can access the disk locally because they are connecting it. When CURLY accesses the disk, however, it arbitrarily chooses a path through either MOE or LARRY.

If CURLY tried to access the disk by using the conventional cluster device name, MOE$DBA1, and a failure had occurred on node MOE before CURLY had booted, access to the disk would fail because the path through MOE would be unknown to CURLY. (Note, however, that if the failure had occurred on MOE after CURLY booted, access to MOE$DBA1 would be granted through LARRY.) By naming the disk independently of the node (path), however, access to the disk can be made without dependence on the availability of any single path.

Figure 4–5 Cluster Configuration With a Dual-Pathed Disk



You assign a dual-pathed disk a unique, path-independent device name by assigning the nodes connecting the device a value called an *allocation class*. The allocation class is a numeric value from 0 through 255.

The allocation class is used to create a device name in the form

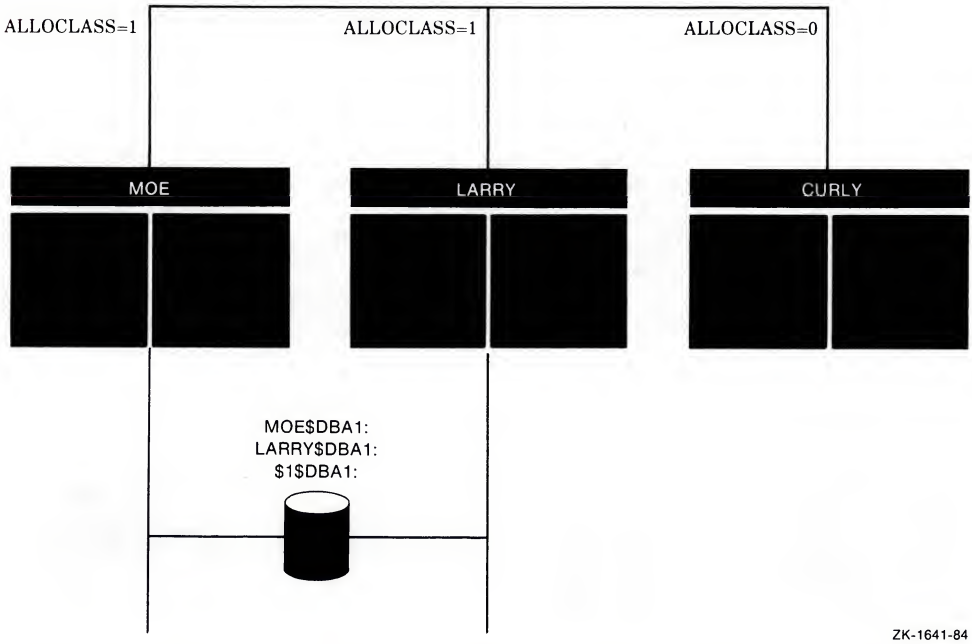
```
$allocation-class$device-name
```

For example, the following allocation class device name identifies a disk that is dual ported between two HSC50s that each have an allocation class of 10:

```
$10$DUA6
```

Figure 4–6 depicts the same configuration shown in Figure 4–5, but includes the conventional device names and the allocation class values for nodes MOE and LARRY with the resulting allocation class device name. Access to the dual-pathed disk by node CURLY is directed to 1DBA1:; the path to it is chosen arbitrarily.

Figure 4–6 Cluster Configuration With a Named Dual-Pathed Disk



To assign an allocation class value to a VAX node that supports dual-pathed devices, specify an allocation class value with the SYSGEN parameter ALLOCLASS (see Chapter 5). The ALLOCLASS value specifies an allocation class for VAX nodes running VAX/VMS.

To assign an allocation class for an HSC, specify the allocation class value using the HSC50 console to enter the command below, where n is the allocation class number.

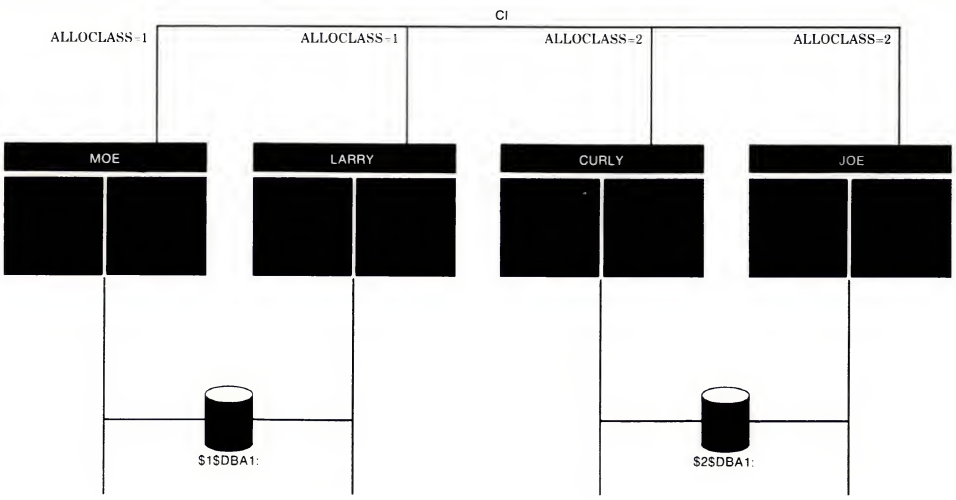
```
SET ALLOCATE DISK n
```

Note that 0, which is the default value for ALLOCLASS, is a special allocation class value. Any node in the cluster that is not connecting a dual-pathed disk should be assigned an allocation class of 0.

Allocation classes play an important role in determining strategies for configurating and naming disks. In fact, VAX/VMS uses allocation class information above all other available information when determining the configuration of devices in the cluster.

For example, the use of allocation classes allows two disks having the same controller and unit number to be dual-pathed between separate sets of nodes in a cluster. Figure 4–7 illustrates such a configuration.

Figure 4–7 Allocation Classes in Cluster Configuration With Two Dual-Pathed Disks



By assigning a unique allocation class to each set of connecting nodes, each dual-ported disk is given a unique device name. Nodes MOE and LARRY have the allocation class 1, and

therefore the disk dual-pathed between them has a device name of 1DBA1. Because nodes CURLY and JOE have the allocation class 2, the disk dual-ported between them has the device name 2DBA1.

The following rules apply to the use of allocation classes:

- The nodes (VAX or HSC50) connecting a dual-pathed disk must have the same nonzero allocation class.
- All cluster-accessible disks on nodes with a nonzero allocation class must have unique names. For example, suppose nodes MOE and LARRY have the allocation class 3, and all local disks are served by the MSCP server. It is invalid for MOE and LARRY to each have a separate disk named DBA0. This example also applies to HSC50s.
- Restricted-access disks that are not cluster accessible can have the same name on nodes having the same allocation class. For example, nodes MOE and LARRY with an allocation class of 2 can each have a local disk named DBA0 as long as the disks are not cluster accessible (dual ported or MSCP served).

Failure to correctly set the allocation class can result in corrupted disks and allocation or locking conflicts that can suspend system activity.

4.3 Shared Disk Volumes

The ability of nodes to share the same disk volume and the files on that volume is one of the most important features of VAXclusters. A *shared disk volume* is a volume that is mounted on a cluster-accessible device by one or more nodes in the cluster. Shared disk volumes offer several advantages:

- Applications can be distributed across multiple VAX systems and still share a common database.
- Use of mass storage is more efficient because more than one node uses the same disk.
- Nodes can share common versions of files.
- Updates to common files are made once to a single copy of the file.

- Users can access the same default disk and directory when logging in to any node sharing the disk.
- Nodes can share a common job controller queue file, and thus can share queues.

Shared disk volumes play a key role in homogeneous clusters, where certain (or all, if multiple systems are run from a common system disk) system files and system command procedures are set up as common files. By storing these files on a shared disk volume, nodes can share a single copy of each common file (see Chapter 2).

One disadvantage to using shared disk volumes is that a shared disk volume is a single point of failure for data access by the nodes sharing the volume.

To mount cluster-accessible disks that are to be shared among all cluster nodes, specify the same MOUNT command on each node or specify the MOUNT command with the /CLUSTER qualifier on one node. When you execute MOUNT/CLUSTER on one node, the disk is mounted on every node in the cluster at the time the command is issued. Note that only system or group volumes can be mounted cluster-wide; thus, if you specify MOUNT/CLUSTER without the /SYSTEM or /GROUP qualifier, /SYSTEM is assumed. Also note that each volume in a VAXcluster mounted with either the /SYSTEM, /GROUP, or /SHARED qualifier must have a unique volume label.

If you want to mount a shared disk volume on some but not all of the nodes in the cluster, execute the same MOUNT command (without the /CLUSTER qualifier) on each node sharing the volume.

For example, suppose you want all the nodes in a three-node cluster to share a disk volume named COMPANYDOCS. To share the volume, each of the three nodes could execute identical MOUNT commands, or one of the three nodes could mount COMPANYDOCS using the MOUNT/CLUSTER command, as follows:

```
$ MOUNT/SYSTEM/CLUSTER/NOASSIST $2$DUA4 COMPANYDOCS
```

If you want just two of the three nodes to share the volume, the two nodes both must mount the disk using the same MOUNT command. For example:

```
$ MOUNT/SYSTEM/NOASSIST $2$DUA4 COMPANYDOCS
```

To mount the volume at startup time, include the mount command either in a common command procedure that is invoked at startup time, or in the node-specific startup command procedure.

4.4 Setting Up Cluster Devices

To implement your plans for cluster disks, you should create command procedures that set up cluster-accessible devices and mount private and shared disk volumes.

You may want to include commands that set up and mount cluster disks in a separate command procedure file named, for example, CLUS_MOUNT.COM, that is invoked by SYSTARTUP.COM. Depending on your cluster environment, you can set up your CLUS_MOUNT command procedure in either of the following ways:

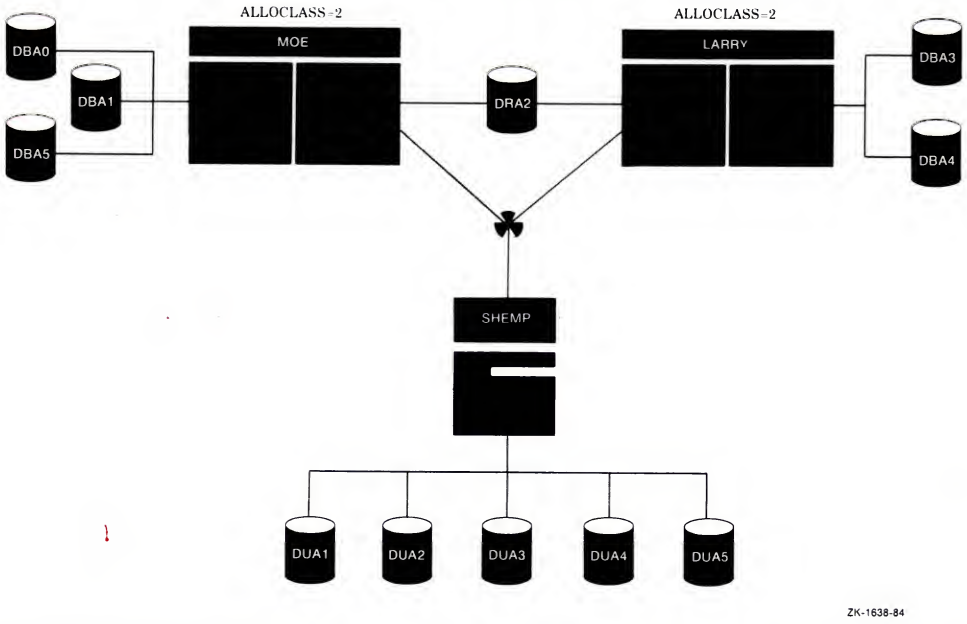
- As a separate file specific to each node in the cluster
- As a common node-independent file

Both methods are described in the following sections. Figure 4–8 illustrates a sample disk configuration for a VAXcluster. In this figure, MOE and LARRY are VAX processors and node SHEMP is an HSC50. The command procedures presented in the following sections demonstrate how to set up cluster devices and mount private and shared disk volumes for this configuration.

4.4.1 Mounting Disks Using Separate Node-Specific Command Procedures

For each node in the cluster, create a CLUS_MOUNT command procedure that is invoked by SYSTARTUP.COM. Each node-specific procedure should contain commands that do the following:

Figure 4–8 Sample Cluster Disk Configuration



1 Add the names of local disks to be shared by way of the MSCP server. Do this by executing the SYSGEN command MSCP followed by the DCL command SET DEVICE/SERVED. (For details, see Section 4.1.2.) This must be done *before* the disk is mounted (the system disk is an exception; it can be served at any time).

2 Mount any local disks. If you served local disks and you want to make those disks accessible to all the nodes in the cluster, mount them using the MOUNT/CLUSTER command (see Section 4.3).

3 Mount HSC disks and remote disks (disks that other nodes added to the MSCP server database).

Example 4–1 demonstrates the use of separate node-specific command procedures to set up and mount shared disks.

Example 4–1 Mounting Disks Using Separate Node-Specific Command Procedures

Command Procedure for Node MOE

```
$ SET NOON
$ !
$ ! Add names to MSCP Server database.
$ !
$ RUN SYS$SYSTEM:SYSGEN
  MSCP
  EXIT
$ SET DEVICE/SERVED MOE$DBA0
$ SET DEVICE/SERVED MOE$DBA1
$ SET DEVICE/SERVED/DUAL $2$DRA2
$ !
$ !
$ ! Mount all local disks.
$ !
$ MOUNT/SYSTEM/CLUSTER MOE$DBA0 MOE16JUNE
$ MOUNT/CLUSTER/SYSTEM MOE$DBA1 USERPACK1
$ MOUNT/CLUSTER/SYSTEM $2$DRA2 USERPACK2
$ !
$ !  Mount HSC disks.
$ !
$ MOUNT/SYSTEM/NOASSIST SHEMP$DUA1 WORK01
$ MOUNT/SYSTEM/NOASSIST SHEMP$DUA2 WORK02
$ MOUNT/SYSTEM/NOASSIST SHEMP$DUA3 WORK03
$ MOUNT/SYSTEM/NOASSIST SHEMP$DUA4 WORK04
$ MOUNT/SYSTEM/NOASSIST SHEMP$DUA5 DOCUMENT
$ !
$ !  Mount remote disks.
$ !
$ MOUNT/SYSTEM/NOASSIST LARRY$DBA3 LARRYPACK
$ MOUNT/SYSTEM/NOASSIST LARRY$DBA4 LARRYWORK
$ !
$ EXIT
```

(Continued on next page)

Example 4–1 (Cont.) Mounting Disks Using Separate Node-Specific Command Procedures

Command Procedure for Node Larry

```
$ SET NOON
$ !
$ ! Add Names to MSCP Server database.
$ !
$ RUN SYS$SYSTEM:SYSGEN
  MSCP
  EXIT
$ SET DEVICE/SERVED LARRY$DBA3
$ SET DEVICE/SERVED LARRY$DBA4
$ SET DEVICE/SERVED/DUAL $2$DRA2
$ !
$ !
$ ! Mount all local disks.
$ !
$ MOUNT/CLUSTER/SYSTEM/NOASSIST LARRY$DBA3 LARRYPACK
$ MOUNT/CLUSTER/SYSTEM/NOASSIST LARRY$DBA4 LARRYWORK
$ MOUNT/CLUSTER/SYSTEM/NOASSIST $2$DRA2 USERPACK2
$ !
$ !  Mount HSC Disks.
$ !
$ MOUNT/SYSTEM/NOASSIST SHEMP$DUA1 WORK01
$ MOUNT/SYSTEM/NOASSIST SHEMP$DUA2 WORK02
$ MOUNT/SYSTEM/NOASSIST SHEMP$DUA3 WORK03
$ MOUNT/SYSTEM/NOASSIST SHEMP$DUA4 WORK04
$ MOUNT/SYSTEM/NOASSIST SHEMP$DUA5 DOCUMENT
$ !
$ !  Mount remote disks.
$ !
$ MOUNT/SYSTEM/NOASSIST MOE$DBA1 USERPACK1
$ !
$ ! EXIT
```

The commands in Example 4–1 perform the following operations for each node:

- Add local disk names to the MSCP server database
- Mount volumes on locally served disks
- Mount volumes that are to be shared on HSC disks
- Mount shared volumes on all MSCP-served disks from other active nodes

Example 4–1 also illustrates the designation of a restricted-access disk. In the example, the names of local disks MOE$DBA1 and 2DRA2 on node MOE, and local disks LARRY$DBA3, LARRY$DBA4, and 2DRA2 on node LARRY are added to the MSCP server database and are shared by the nodes that mount them.

4.4.2 Mounting Disks Using a Common Command Procedure

You can also create a common version of the CLUS_MOUNT command procedure that is node independent. You can set up the common procedure as a shared file on a shared disk, or you can make duplicate copies of the common procedure and store them as separate files.

Using either method, each node can invoke the common version CLUS_MOUNT.COM by invoking the file from SYSTARTUP.COM. Example 4–2 shows a sample common command procedure used to mount cluster disks. (For more information on techniques for mounting cluster disks, see the examples in the SYS$EXAMPLES: directory on your system.)

Example 4–2 Mounting Disks Using a Common Command Procedure

```
$ !
$ SET NOON
$ !
$ SET DEVICE/DUAL_PORTED DRA2
$ !
$ IF F$GETSYI("NODENAME") .NES. "MOE" -
  THEN GOTO NOT_MOE1:
$ !
$ ! Add names of MOE's disks to the MSCP Server database.
$ !
$ RUN SYS$SYSTEM:SYSGEN
  MSCP
  EXIT
$ SET DEVICE/SERVED MOE$DBA0
$ SET DEVICE/SERVED MOE$DBA1
$ SET DEVICE/SERVED/DUAL $2$DRA2
$ !
$ NOT_MOE1:
$ IF F$GETSYI("NODENAME") .NES. "LARRY" -
  THEN GOTO NOT_LARRY1
$ !
$ ! Add names of LARRY's disks to the MSCP Server database.
$ !
$ RUN SYS$SYSTEM:SYSGEN
  MSCP
  EXIT
```

(Continued on next page)

Example 4–2 (Cont.) Mounting Disks Using a Common Command Procedure

```
$ SET DEVICE/SERVED LARRY$DBA3
$ SET DEVICE/SERVED LARRY$DBA4
$ SET DEVICE/SERVED/DUAL $2$DRA2
$ !
$ NOT_LARRY1:
$ !
$ ! Mount the restricted-access disk.
$ !
$ IF F$GETSYI("NODENAME") .EQS. "MOE" THEN -
  MOUNT/SYSTEM/NOASSIST MOE$DBA0 MOE16JUN
$ !
$ ! Mount MSCP served disks.
$ !
$ MOUNT/CLUSTER/SYSTEM/NOASSIST  MOE$DBA1 USERPACK1
$ MOUNT/CLUSTER/SYSTEM/NOASSIST  $2$DRA2 USERPACK2
$ MOUNT/CLUSTER/SYSTEM/NOASSIST  LARRY$DBA3 LARRYPACK
$ MOUNT/CLUSTER/SYSTEM/NOASSIST  LARRY$DBA4 LARRYWORK
$ !
$ ! Mount HSC disks.
$ !
$ MOUNT/SYSTEM/NOASSIST SHEMP$DUA1 WORK01
$ MOUNT/SYSTEM/NOASSIST SHEMP$DUA2 WORK02
$ MOUNT/SYSTEM/NOASSIST SHEMP$DUA3 WORK03
$ MOUNT/SYSTEM/NOASSIST SHEMP$DUA4 WORK04
$ MOUNT/SYSTEM/NOASSIST SHEMP$DUA5 DOCUMENT
```

The command procedure in Example 4–2 performs the same mount operations as the command procedures shown in Example 4–1. The command procedure in Example 4–2, however, executes a common set of commands that mount the same disks for each node that executes the procedure.

5 Forming the Cluster

After you have prepared the operating environment, you are ready to form the cluster. A VAXcluster does not exist until a VAX/VMS node with the necessary system parameters set is booted. Booting the first node creates the cluster. Any nodes that boot after the cluster is formed are said to join the cluster.

This chapter discusses the cluster software that dynamically coordinates cluster membership and describes cluster system parameters and the procedures required to form, maintain, and shut down a VAXcluster.

5.1 Cluster Connection Management

The VAXcluster is dynamically controlled by a software component called the *connection manager*, which determines cluster membership and provides the coordination needed to manage it. It is the connection manager, for example, that creates a cluster when the first active node is booted and reconfigures the cluster when nodes join or leave it.

A list of the connection manager messages displayed at the operator terminals throughout the cluster is included in Appendix B.

In a VAXcluster, member nodes can share various data and system resources, such as disk volumes. The integrity of shared resources, however, cannot be guaranteed unless the use of shared resources is carefully coordinated in the cluster. In the unlikely event that a pair of nodes share some resource, but are not members of the same cluster and therefore cannot coordinate the use of that resource, the integrity of the shared resource is not assured.

To achieve the coordination necessary to maintain resource integrity, the nodes in a VAXcluster must share a clear sense of cluster membership. This sense of cluster membership is maintained by the connection manager, which prevents another cluster from sharing the same resources.

Two clusters sharing the same resources is called a *partitioned cluster*. Cluster partitioning occurs when two active nodes that are intended to be members of the same cluster operate independently as members of two different clusters. Partitioning is undesirable because resource sharing between two clusters is not coordinated. The connection manager prevents cluster partitioning using a scheme called *quorum*.

5.1.1 The Quorum Scheme

The quorum scheme is based on the arithmetic fact that the whole cannot be divided into multiple parts in such a way that more than one part is greater than half of the whole.

The quorum scheme functions as follows:

1. Each VAX node in the cluster contributes a fixed number of votes toward a quorum. Each node specifies a votes value using the SYSGEN parameter VOTES.
2. The connection manager dynamically computes the cluster votes value as the sum of the votes held by member nodes.
3. Each active node in the cluster specifies an initial quorum value using the SYSGEN parameter QUORUM. This parameter is an estimate of the correct quorum value for the cluster.
4. During certain cluster state transitions, the system dynamically computes the cluster quorum to be the *maximum* of the following:
   - The current cluster quorum value
   - The value for quorum specified by each node
   - The value calculated from the following formula, where V is the cluster votes total.

     ```
     (V+2)/2
     ```

   The cluster state transitions that cause cluster quorum to be recalculated occur when a node joins the cluster and when the cluster recognizes a quorum disk (see Section 5.1.2).

5 If the current number of votes ever drops below the quorum (due to nodes leaving the cluster), the cluster members suspend all process activity and all I/O to cluster-accessible disks until sufficient votes are added (nodes joining the cluster) to bring the total number of votes to a value greater than or equal to quorum.

6 As the cluster changes, the system only raises the cluster quorum value; it never lowers the value. (However, you as cluster manager can lower the value; for details, see Section 5.4.)

Consider a cluster consisting of three nodes, each node having its VOTES parameter set to 1 and its QUORUM parameter set to 2. The cluster dynamically computes the cluster votes value to be 3 and the quorum value to be 2. In this example, any two of the three nodes constitute a quorum and may run in the absence of the third node. No single node can constitute a quorum by itself. Therefore, there is no way the three cluster nodes can be partitioned and run as two independent clusters.

5.1.2 Quorum Disk

If you have a two-node cluster, you can increase its availability by establishing a quorum disk. A quorum disk acts as a virtual node in the cluster, adding votes (the minimum of the SYSGEN parameter QDSKVOTES settings on all nodes that have been cluster members) to the cluster votes total. For the quorum disk's votes to be counted in the cluster votes total, the following conditions must be met:

- On each cluster node you must specify the name of the quorum disk by setting the SYSGEN parameter DISK_QUORUM. The DISK_QUORUM value must be the same on each node.
- The specified disk must be accessible by every node in the cluster.
- The disk must contain a valid format file named QUORUM.DAT in the master file directory (MFD). The QUORUM.DAT file is created automatically after a system specifying a quorum disk has booted and run as a cluster member. On the initial booting of a system, the file will not be present; therefore, provisions for a quorum without the disk must be made.

When a quorum disk contributes to the cluster votes total, a two-node cluster with a shared HSC50 or MASSBUS disk can tolerate the failure of either one VAX node or the quorum disk.

Note that clusters with more than two VAX nodes may also use a disk as a virtual node; however, if your cluster is large enough so that a quorum disk would not significantly improve cluster availability, DIGITAL recommends that you do not specify a quorum disk.

5.2 VAXcluster SYSGEN Parameters

For systems to boot properly into a VAXcluster, certain VAXcluster system parameters must be set on each cluster node. Table 5–1 lists the SYSGEN parameters required by VAXcluster nodes.

Table 5–1 Cluster System Parameters

Parameter	Description
Cluster Parameters	
ALLOCLASS	Specifies a numeric value to be assigned as the allocation class for the node.
DISK_QUORUM	The name, in ASCII, of an optional quorum disk. ASCII spaces indicate that no quorum disk is being used.
QDSKVOTES	Specifies the number of votes contributed to the cluster votes total by a quorum disk. The maximum is 127, the minimum is 0, and the default is 1.
QDISKINTERVAL	Specifies the disk quorum polling interval, in seconds. The maximum value is 32,767, the minimum value is 1, and the default is 20. Lower values trade increased overhead cost for greater responsiveness. DIGITAL recommends that this parameter be set to the same value on each cluster node.

Table 5–1 (Cont.) Cluster System Parameters

Parameter	Description
QUORUM	Specifies an initial setting for the dynamic quorum value. This setting is a numeric value that is an estimate of the correct quorum value to be used and should be greater than half of the total expected votes. By default, the value is 1. The QUORUM value should be set to a number that will prevent the partitioning of a cluster (see Section 5.4). The system uses the formula (V+2)/2, where V is the sum of the votes held by the nodes of your VAXcluster (including the quorum disk, if one is used) to calculate quorum.
RECNXINTERVAL	Specifies, in seconds, the interval during which the connection manager attempts to reconnect a broken connection to another VAX/VMS system. If a new connection cannot be established during this period, the connection is declared irrevocably broken, and either this system or the other must leave the cluster. This parameter trades faster response to certain types of system failures against the ability to survive transient faults of increasing duration. DIGITAL recommends that this parameter be set to the same value on each cluster node.

Table 5–1 (Cont.) Cluster System Parameters

Parameter	Description
VAXCLUSTER	Controls whether the system should join or form a VAXcluster. This parameter accepts the following three values: • 0—Specifies that the system will not participate in a VAXcluster. • 1—Specifies that the system should participate in a VAXcluster if hardware supporting SCS is present (CI, UDA, HSC50). • 2—Specifies that the system should participate in a VAXcluster You should always set this parameter to 2 on systems intended to run in a VAXcluster, 0 on systems that boot from a UDA and are not intended to be part of a VAXcluster, and 1 (the default) otherwise.
VOTES	Specifies the number of votes towards a quorum to be contributed by the node. By default, the value is 1.
SCS Parameters	
PANUMPOLL	Specifies the number of ports to poll at each interval. DIGITAL recommends that this parameter be set to the same value on each cluster node.
PASTIMOUT	Specifies the interval at which the CI port driver performs time-based bookkeeping operations. This interval is also the period after which a start handshake datagram is assumed to have timed out. Normally the default value is adequate. DIGITAL recommends that this parameter be set to the same value on each cluster node.

Table 5–1 (Cont.) Cluster System Parameters

Parameter	Description
PASTDGBUF	Specifies the number of datagram receive buffers to queue for the CI port driver's configuration poller; that is, the maximum number of start handshakes that can be in progress simultaneously. Normally the default value is adequate. DIGITAL recommends that this parameter be set to the same value on each cluster node.
PAMAXPORT	Specifies the maximum number of CI ports the CI port driver polls for a broken port-to-port virtual circuit, or a failed remote node. You can decrease this parameter in order to reduce polling activity if the hardware configuration has fewer than 16 ports. For example, if the configuration has a total of five ports assigned port numbers 0-4, then you should set PAMAXPORT to 4. The default for this parameter is 15 (poll for all possible ports 0 through 15). DIGITAL recommends that this parameter be set to the same value on each cluster node.
PANOPOLL	Disables polling if set to 1. (The default is 0.) Disabling polling enables you to boot a system from a local system disk and isolate it from CI activity. You may want to do this following repairs to verify that the system runs properly before introducing it into the hardware cluster. Never set PANOPOLL to 1 under these conditions: a system is participating in a cluster, a system is being booted from an HSC, or a system is being booted in order to join a cluster.

Table 5–1 (Cont.) Cluster System Parameters

Parameter	Description
PAPOLLINTERVAL	Specifies in seconds, the polling interval the computer interconnect (CI) port driver uses to poll for a newly booted system, a broken port-to-port virtual circuit, or a failed remote node. This parameter trades polling overhead against quick response to virtual circuit failures. DIGITAL recommends that you use default value for this parameter. DIGITAL recommends that this parameter be set to the same value on each cluster node.
PAPOOLINTERVAL	Specifies in seconds, the interval at which the PA port driver checks for available nonpaged pool after a failure to allocate. Normally the default value is adequate.
PASANITY	Controls whether the port sanity timer is enabled to permit remote systems to detect a system that has been halted or retained at IPL 7 or above for 99 seconds. This parameter is normally set to 1 and should only be set to 0 when debugging with XDELTA. PASANITY is a dynamic parameter (altered the next time the port is initialized) and has a default value of 1.
PRCPOLINTERVAL	Specifies, in seconds, the polling interval used to look for SCS applications, such as the connection manager and MSCP disks, on other nodes. Each node is polled, at most, once each interval. This parameter trades polling overhead against quick recognition of new systems or servers as they appear. DIGITAL recommends that you set this parameter to 15, which is the default.
SCSBUFFCNT	Specifies the number of computer interconnect (CI) buffer descriptors configured for all CI ports on the system.

Table 5–1 (Cont.) Cluster System Parameters

Parameter	Description
SCSCONNCNT	Specifies the total number of SCS connections that are configured for use by all system applications. Normally, the default value is adequate.
SCSFLOWCUSH	Specifies the lower limit for receive buffers at which point SCS starts to notify the remote SCS of new receive buffers. For each connection, SCS tracks the number of receive buffers available. SCS communicates this number to the SCS at the remote end of the connection. However, SCS does not need to do this for each new receive buffer added. Instead, SCS notifies the remote SCS of new receive buffers if the number of receive buffers falls as low as the SCSFLOWCUSH value. Normally the default value is adequate.
SCSSYSTEMID	Specifies the lower-order 32 bits of the 48-bit system identification number. This parameter is not dynamic and must be the same as the DECnet node number (1024 * <DECnet area> + DECnet node number).
SCSSYSTEMIDH	Specifies the high-order 16 bits of the 48 bit system identification number. This parameter must be set to 0. It is reserved by DIGITAL for future use.
SCSNODE	Specifies the SCS system name. This parameter is not dynamic. You should use a name that is the same as the DECnet node name (limited to six characters) since the name must be unique among all systems in the cluster. Note that once a node has been recognized by another node in the cluster, you cannot change the SCSSYSTEMID or SCSNODE parameter without changing both.
SCSRESPCNT	Specifies the total number of response descriptor table entries configured for use by all system applications.

5.3 Booting Nodes in a VAXcluster

You form the cluster by setting the VAXcluster SYSGEN parameters to the appropriate values and booting each node.

There are two ways you can set VAXcluster system parameters before you boot a node into the cluster. The method that you choose is a matter of preference since each method accomplishes the same results.

- Before shutting down a node that is running as a single-node system, run SYSGEN to set the required VAXcluster parameters. Then, you can modify the current system SYSGEN parameters (SYS$SYSTEM:VAXVMSSYS.PAR) by first issuing a USE CURRENT command, modifying the desired parameters, and then using the SYSGEN command WRITE CURRENT. When you reboot the node, it will join the cluster with the necessary SYSGEN parameters in place.

- Shut down and reboot each node using the conversational bootstrap procedure. When the procedure stops in SYSBOOT, set the necessary SYSGEN parameters by entering the proper SYSGEN commands at the SYSBOOT prompt. The *Guide to VAX/VMS System Management and Daily Operations* describes the conversational bootstrap procedure.

To form the VAXcluster using the conversational bootstrap method, perform the following steps for each node joining the cluster:

1 Boot the system using the conversational bootstrap procedure described in the *Guide to VAX/VMS System Management and Daily Operations*.

2 When the boot procedure is ready to accept commands, it pauses and the following command prompt is displayed:

```
SYSBOOT>
```

At the prompt, use the SET command to establish the values of the following VAXcluster SYSGEN parameters:

- ALLOCLASS
- DISK_QUORUM
- QUORUM
- SCSSYSTEMID
- SCSSYSTEMIDH
- SCSNODE
- VAXCLUSTER
- VOTES

3 Resume the bootstrap operation by specifying the CONTINUE command.

Once you have performed these steps on each cluster node, run AUTOGEN and reboot each system, one at a time. The cluster will form when enough nodes have been booted to attain cluster quorum.

5.4 Maintaining the Cluster

During the life of a VAXcluster, nodes join and leave the cluster. For example, you may need to add more processors to the cluster to extend the cluster's processing capabilities, or a node in the cluster may shut down unexpectedly due to a hardware or fatal software error. The connection management software coordinates these cluster transitions and controls cluster operation.

When a cluster node shuts down unexpectedly, the remaining nodes, with the help of the connection manager, reconfigure the cluster, excluding the node that shut down. The cluster will survive the failure of the node and continue to process, as

long as the cluster votes total is greater than the cluster quorum value. If the cluster votes total falls below the cluster quorum value, the cluster suspends the execution of all processes.

For process execution to resume, the cluster votes total must be restored to a value greater than or equal to the cluster quorum value. Often, the required votes are added as nodes join or rejoin the cluster. However, waiting for a node to join the cluster and raise the votes value is not always a simple or convenient remedy. An alternative solution, for example, might be to shut down and reboot all the nodes with a lower quorum value. In any case, it is important to be aware of cluster state changes in order to prevent potential problems.

Following the failure of a node, you may wish to run the Show Cluster Utility and examine values for the VOTES, QUORUM, CL_VOTES, and CL_QUORUM fields. (See the *VAX/VMS Utilities Reference Volume* for a complete description of the Show Cluster Utility.) The VOTES and QUORUM fields show the votes and quorum settings for each cluster member; the CL_VOTES and CL_QUORUM fields show the cluster votes total and the current cluster quorum value.

To examine these values, specify the following commands:

```
$ SHOW CLUSTER/CONTINUOUS
COMMAND> ADD VOTES,QUORUM,CL_VOTES,CL_QUORUM
```

Note: You must specify the /CONTINUOUS qualifier as part of the SHOW CLUSTER command string, if you wish to interactively invoke the Show Cluster Utility. If you do not specify this qualifier, you will receive the display of cluster status information returned by the DCL command SHOW CLUSTER and you will remain at the DCL level.

If the display from the Show Cluster Utility shows the CL_VOTES value equal to the CL_QUORUM value, the cluster will not survive the failure of any one of the remaining nodes. If one of the remaining nodes shuts down, all process activity on the cluster will stop.

You can reduce this vulnerability and prevent the disruption of cluster process activity by lowering the cluster quorum value. The DCL command SET CLUSTER/QUORUM allows you to manually adjust the cluster quorum to a value you specify or, if no value is specified, to a value calculated by a system formula.

If, for example, you wish to set the quorum to a value of 3, enter the following command string:

```
$ SET CLUSTER/QUORUM=3
```

If you enter SET CLUSTER/QUORUM and do not specify a value for quorum, the system calculates the value for you, using the formula below, where V is the cluster votes total.

```
(V+2)/2
```

Note that no matter what value you specify as part of the SET CLUSTER/QUORUM command, you cannot increase quorum to a value that is greater than the number of the votes present, nor can you reduce quorum to a value that is half or less of the votes present.

When you issue the SET CLUSTER/QUORUM command, either with or without a quorum value specified, the system will respond with a message indicating the new quorum value that was actually set.

You only need to enter the SET CLUSTER/QUORUM command on one node to change the cluster quorum, since that value will be propagated throughout the cluster. After you change the quorum, however, you should store the new quorum value in the SYSGEN parameter QUORUM on each cluster node, so that it remains in effect after the nodes reboot.

Note: To use the SET CLUSTER/QUORUM command, you must have OPER (operator) privilege.

When a node that was previously a member of the cluster is ready to rejoin, you should increase the SYSGEN parameter QUORUM to its original value on all nodes. Note that you do not need to use the SET CLUSTER/QUORUM command to increase the cluster quorum, since the quorum value will be increased automatically when the node rejoins the cluster.

Note that in general, you only need to use SET CLUSTER /QUORUM when a node is leaving the cluster for an extended period of time. For more information on the DCL command SET CLUSTER/QUORUM, see the description in the *VAX/VMS DCL Dictionary*.

You can also reduce cluster quorum by selecting one of the cluster-related shutdown options. These options are described in the next section.

5.5 Shutting Down the Cluster

The VAX/VMS operating system provides three options for shutting down nodes in the VAXcluster:

- REMOVE_NODE
- CLUSTER_SHUTDOWN
- REBOOT_CHECK

Depending on which option you choose, you can either shut down individual nodes in a cluster without bringing down the entire cluster or shut down the entire cluster.

If you do not select any of the above options when shutting down a node in your cluster, the shutdown procedure will default to the normal behavior applicable to shutting single node systems. If you wish to shut down a node in your cluster that you expect will be rejoining the cluster shortly, you can use the normal shutdown procedure. Note that in the case of the normal shutdown operation, cluster quorum will not be adjusted, since it is assumed that the node will soon be rejoining the cluster.

If you wish to shut down a certain node in your cluster that you expect will not be rejoining the cluster for an extended period of time, select the REMOVE_NODE option in your SHUTDOWN procedure. For example, a node in your cluster may be waiting for new hardware, or you may decide that you want to use a certain node in your cluster as a stand-alone node for an extended period of time.

When you use the REMOVE_NODE option, the active quorum in the remainder of the cluster will be adjusted downward to reflect the fact that the removed node's votes will no longer be contributing to the quorum value. The SHUTDOWN procedure readjusts the quorum by issuing the SET CLUSTER/QUORUM command, which is subject to the usual constraints described in Section 5.4.

Note that it is still the responsibility of the cluster manager to change the SYSGEN parameter QUORUM in the remaining nodes, to reflect the new configuration. This value should be set according to standard formula, (V+2)/2, described above. This value can also be taken from any remaining cluster member after the node being removed has halted by examining the current quorum value on the remaining nodes. You can also do this with the lexical function F$GETSYI as follows:

```
$ Q = F$GETSYI("CLUSTER_QUORUM")
$ SHOW SYMBOL Q
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> USE CURRENT
SYSGEN> SET QUORUM q     ! value displayed by SHOW SYMBOL
SYSGEN> WRITE CURRENT
SYSGEN> EXIT
```

If you wish to shut down the entire cluster, select the CLUSTER_SHUTDOWN option in your SHUTDOWN procedure. When you select this option, the node will suspend activity, just short of shutting down completely, until all nodes in the cluster have reached the same point in the shutdown procedure. When this happens, all nodes in the cluster will shut down together.

Note that when you select the CLUSTER_SHUTDOWN option to perform a cluster-wide shutdown operation, you must still shut down each node in the cluster by invoking the SHUTDOWN.COM procedure at each node's console. If any one node in the cluster is not completely shut down, cluster-wide shutdown cannot occur. Instead, all of the other nodes in the cluster will remain suspended.

When shutting down nodes in the cluster, you can also select the REBOOT_CHECK option. When you select this shutdown option, the procedure will check for the existence of basic system files that are needed to successfully reboot the system and notify you if any files are missing. You should replace such files before proceeding. If all files are present, you will receive the following success message:

```
%SHUTDOWN-I-CHECKOK, Basic reboot consistency check completed.
```

Note that you can select the REBOOT_CHECK option separately or in conjunction with either the REMOVE_NODE or CLUSTER_SHUTDOWN option. If you select REBOOT_CHECK with one of the other options, be sure to separate the option list with a comma.

A Building a Common SYSUAF.DAT File on Upgraded Systems

This appendix contains guidelines for building a user authorization file on a system that has been upgraded. For more detailed information on how to set up a single user authorization file on a local node, see the descriptions in the Authorize Utility (AUTHORIZE) in the *VAX/VMS Utilities Reference Volume* and in the *Guide to VAX/VMS System Management and Daily Operations.*

To build a common SYSUAF.DAT file on upgraded systems, perform the following steps.

1 Boot each cluster node in the single-system environment (see Section 2.2) and print a listing of SYSUAF.DAT. To print this listing, invoke AUTHORIZE and specify the AUTHORIZE command LIST as follows:

```
$ SET DEF SYS$SYSTEM
$ RUN AUTHORIZE
UAF> LIST/FULL [*,*]
```

2 Use the listings to compare the accounts from each node. On the listings, mark down any necessary changes.

One such change is to delete any accounts that you no longer need. You should also make sure that each user account in the cluster has a unique UIC.

For example, node BLUE of the cluster may have a user account JONES that has the same UIC as user account SMITH on node GREEN. When nodes BLUE and GREEN are joined to form a cluster, accounts JONES and SMITH will exist in the cluster environment with the same UIC. If the UICs of these accounts are not differentiated, each user will have the same access rights to various objects in the cluster. In this case you should assign each account a unique UIC.

Another consideration is to make sure that accounts that perform the same type of work have the same group UIC. Accounts in a single-system environment probably follow this convention. However, there may be groups of users on each node that will perform the same work in the cluster but have group UICs unique to their local node. As a rule, the group UIC for any given work category should be the same on each node in the cluster. For example, data entry accounts on node BLUE should have the same group UIC as data entry accounts from node GREEN and node RED.

Note that if you change the UIC for a particular user, you should also change the owner UICs for that user's existing files and directories. You can use the DCL commands SET FILE and SET DIRECTORY to make these changes. These commands are described in detail in the *VAX/VMS DCL Dictionary*.

3 Choose the SYSUAF.DAT from one of the nodes to be a master SYSUAF.DAT.

4 Merge the SYSUAF.DAT files from the other nodes to the master SYSUAF.DAT by running the Convert Utility (CONVERT) on the node that owns the master SYSUAF.DAT. (See the *VAX/VMS Utilities Reference Volume* for a description of CONVERT.) To use CONVERT to merge the files, each SYSUAF.DAT file must be accessible to the node that is running CONVERT.

To merge the UAFs into the master SYSUAF.DAT file, specify the CONVERT command in the following format:

```
$ CONVERT SYSUAF1,SYSUAF2,...SYSUAFN MASTER_SYSUAF
```

Note that if a given user name appears in more than one source file, only the first occurrence of that name will appear in the merged file.

The command in the following example adds the SYSUAF.DAT file from two cluster nodes to the master SYSUAF.DAT in the current default directory:

```
$ SET DEFAULT SYS$SYSTEM
$ CONVERT [SYS1.SYSEXE]SYSUAF.DAT,
[SYS2.SYSEXE]SYSUAF.DAT SYSUAF.DAT
```

The CONVERT command in this example adds the records from the files [SYS1.SYSEXE]SYSUAF.DAT and [SYS2.SYSEXE]SYSUAF.DAT to the file SYSUAF.DAT on the local node.

After you run CONVERT, you are left with a master SYSUAF.DAT that contains records from the other SYSUAF.DAT files.

5 Use AUTHORIZE to modify the accounts in the master SYSUAF.DAT according to the changes you marked on the initial listings of the SYSUAF.DAT files from each node. See the description of AUTHORIZE in the *VAX/VMS Utilities Reference Volume*.

B Connection Manager Messages

This appendix contains a list of connection manager messages that are displayed at the operator's terminal. These messages provide information from the connection manager about the creation and reconfiguration of the cluster when nodes join or leave the cluster, and they are useful for diagnosing problems in the cluster.

For more information on the connection manager, see Chapter 5.

The following messages are displayed on the console terminal. Similar OPCOM messages are broadcast on those terminals throughout the cluster that have been designated as cluster operator terminals. Note that the connection manager messages will continue to broadcast even if OPCOM is disabled.

%CNXMAN, Discovered system NODE-NAME

> **Explanation:** The connection manager has discovered the existence of a node (NODE-NAME) that has a connection manager.

%CNXMAN, Established connection to system NODE-NAME

> **Explanation:** The connection manager has initiated connection to a newly discovered node.

%CNXMAN, Lost connection to system NODE-NAME

> **Explanation:** The connection manager discovers that its connection to node NODE-NAME is broken.

%CNXMAN, Re-established connection to system NODE-NAME

> **Explanation:** The connection manager has recovered from a transient failure in connection.

%CNXMAN, Timed-out lost connection to system NODE-NAME

Explanation: The connection manager has given up trying to reestablish connection to the lost node. At this point, the connection manager must reconfigure the cluster.

%CNXMAN, Deleting CSB for system NODE-NAME

Explanation: A node in the cluster has rebooted.

%CNXMAN, Proposing formation of a VAXcluster

Explanation: When a node booted it discovered an existing cluster and is proposing the formation of a new cluster.

%CNXMAN, Sending VAXcluster membership request to system NODE-NAME

Explanation: A node has booted, discovered an existing cluster, and is seeking membership in the cluster.

%CNXMAN, Received VAXcluster membership request from system NODE-NAME

Explanation: The connection manager has received the node's request for membership.

%CNXMAN, Proposing reconfiguration of the VAXcluster

Explanation: The node has determined that the cluster needs reconfiguring and that the node will attempt to begin the process of reconfiguration.

%CNXMAN, Proposing modification of quorum or quorum disk membership

Explanation: A node believes that the quorum disk should be added or removed from cluster membership or that the cluster quorum should be readjusted.

%CNXMAN, Proposing addition of system NODE-NAME

Explanation: The connection manager is coordinating the cluster in admitting the node as a member of the cluster.

%CNXMAN, Aborting VAXcluster state transition

Explanation: A proposed state transition cannot be completed and that the request should be aborted.

%CNXMAN, Completing VAXcluster state transition

Explanation: A proposed state transition has successfully completed.

%CNXMAN, Removed from VAXcluster system NODE-NAME

Explanation: A node previously in the cluster has been removed from the cluster. Each remaining node in the cluster prints this message.

%CNXMAN, Now a VAXcluster member—system NODE-NAME

Explanation: A node has been added to to the cluster. Each node in the cluster prints this message.

%CNXMAN, Detected member of another VAXcluster—system NODE-NAME

Explanation: The connection manager has established a connection to a member of another cluster. At least one of the pair of nodes involved must shut down.

%CNXMAN, Quorum lost, blocking activity

Explanation: Quorum has been lost and processing will be suspended until quorum is regained.

%CNXMAN, Quorum regained, resuming activity

Explanation: Quorum has been regained and that processing will resume.

%CNXMAN, Established "connection" to quorum disk

Explanation: The connection manager has succeeded in communicating with the quorum disk.

%CNXMAN, Lost "connection" to quorum disk

Explanation: The connection manager has failed to communicate with the quorum disk.

%CNXMAN, Error reading quorum disk

Explanation: The connection manager is unable to read from the quorum disk.

%CNXMAN, Error writing quorum disk

Explanation: The connection manager is unable to write to the quorum disk.

%CNXMAN, Quorum disk write-locked

Explanation: The quorum disk has become write-locked.

%CNXMAN, Read invalid data from quorum disk

Explanation: The data read from the quorum disk by the connection manager is invalid.

%CNXMAN, Detected another VAXcluster via the quorum disk

Explanation: A VAXcluster has, by way of the quorum disk, discovered another cluster. At least one of the pair of nodes involved must shut down.

%CNXMAN, Timed-out I/O operation to quorum disk

Explanation: The connection manager's I/O requests to the quorum disk did not complete in reasonable amount of time. (The time is given in seconds by the SYSGEN parameter QDSKINTERVAL)

C Booting From a Common System Disk

When you boot from a common system disk, you must alter the appropriate boot command file on the console volume of the HSC50. (You alter CIBOO.CMD, if you want to use a nonstop boot, or CIGEN., if you want to use a conversational boot.) Before booting, however, you must add two pieces of information to the appropriate file:

- The node number of the HSC50
- The device's unit address

Also, when you want to boot your system from an alternate root—that is, a root other than SYS0—you must identify the root.

You provide this information by editing the selected boot command file (CIBOO.CMD or CIGEN.), after you copy it to a disk directory. To copy the file, you can use the VAX/VMS Exchange Utility (EXCHANGE). When you complete the editing, you use EXCHANGE to copy the edited file back to the console volume (assigning it an appropriate file name). If you want to use the file as your default boot command procedure, copy the file to DEFBOO.CMD.

A typical example of how to set up DEFBOO.CMD to do a non-stop system boot from an RA60 assigned unit number 03 on the HSC controller assigned node number 0C is provided below. Note that this example assumes that the system is booted from SYSA.

1 Be sure your console volume is connected. Use the following command:

```
$ SHOW DEVICE CSA1
```

If the console volume is not connected, invoke SYSGEN to connect it using the following command:

```
SYSGEN> CONNECT CONSOLE
```

2 Exit from SYSGEN and mount the console volume (in this example, the console volume is assumed to be physically located in the floppy drive (CSA1) on a VAX–11/780.)

```
$ MOUNT/FOREIGN CSA1:
```

3 Invoke the Exchange Utility to copy CIBOO.CMD to your default disk directory (or whichever directory is most convenient to you), by entering the following command:

```
$ EXCHANGE
```

You will then receive the EXCHANGE prompt:

```
EXCHANGE>
```

4 Enter the following command:

```
EXCHANGE> COPY CSA1:CIBOO.CMD *.*
```

5 Exit from the Exchange Utility.

6 Edit the disk file you named CIBOO.CMD by adding the two lines for R2 and R3, and by modifying the line for R5:

```
DEPOSIT R0 20              !CI PORT DEVICE
DEPOSIT R1 E               !CI TR=E
DEPOSIT R2 0C          !HSC NODE NUMBER
DEPOSIT R3 03          !DISK DEVICE UNIT NUMBER
DEPOSIT R4 0               !BOOT BLOCK LBN (NOT USED)
DEPOSIT R5 A0000000    !ROOT DESIGNATION
```

Note: If you are using a VAX-11/750 processor, you must use the console command format that is compatible with it. For example, to deposit a hex 03 in R3, you use the form:

```
D/G 3 03               !DISK DEVICE UNIT NUMBER
```

7 Invoke the Exchange Utility once again to copy the edited disk file to DEFBOO.CMD:

```
EXCHANGE> COPY device:[directory]CIBOO.CMD CSA1:DEFBOO.CMD
```

If the disk is dual-ported (accessible to two HSC controllers), you should specify the node number of both controllers in register R2. You should also specify the higher-numbered node in bits <15:8> and the lower-numbered node in bits <7:0>. For example, if the disk is accessible to the HSC numbered 0C and the HSC numbered 0A, the line for R2 should appear like this:

```
DEPOSIT R2 0C0A          !DUAL-PORTED HSC NODE NUMBERS
```

When you have copied the file back to the console volume, exit from EXCHANGE. You can then delete the disk file CIBOO.CMD, if you wish. When you invoke the default bootstrap command procedure, it will boot your system from the designated device on the HSC50.

D Cluster Startup Command Files

This appendix contains three examples of command files used in starting up a cluster. The first two examples describe the system specific procedures for starting up nodes MOE and LARRY. The third example includes a cluster-common command file, called by the respective system specific command procedures, for starting up MOE and LARRY.

System-Specific Startup File for Node MOE

```
$!
$! This is the site-specific startup command procedure for MOE
$! Created 31-Au-1984
$!
$!
$! First check to see if it is in a cluster
$ member=f$getsyi("CLUSTER_MEMBER")
$ IF .NOT. MEMBER THEN $EXIT
$
$ @sys$common:[sysmgr]common_startup
$!
$!
$ submit /noprint /que=moe_batch sys$manager:startnet
$!
$ define/sys/exec  wrkd  WORKA:
$!
$ define/sys/exec  sysd  SYS$SYSDISK:
$!
$!  Define the login announcement
$!
$ define/sys/exec sys$announce "Moe is an 11/780!"
$!
$!
$!  Inform users that the system is up and running.
$!
$ reply/all/bell "MOE VAX-11/780 System initialized"
$ EXIT
```

System-Specific Startup File for Node LARRY

```
$!
$! This is the site-specific startup command procedure for LARRY
$! Created 31-Aug-1984
$!
$!
$! First check to see if it is in a cluster
$ member=f$getsyi("CLUSTER_MEMBER")
$ IF .NOT. MEMBER THEN $EXIT
$
$ @sys$common:[sysmgr]common_startup
$ submit /noprint /que=Larry_batch sys$manager:startnet
$!
$ define/sys/exec wrkd  WORKA:
$!
$ define/sys/exec v4text  sys$sysdevice:[notes]vmsv4.txt
$!
$!  Define the login announcement
$!
$ define/sys/exec sys$announce " LARRY is up and running!"
$ define/sys/exec sys$welcome  " Welcome to the VMS_Cluster"
$!
$!
$!  Inform users that the system is up and running.
$!
$ reply/all/bell "VAX-11/750 System initialized"
$ EXIT
```

Common Startup File for Nodes MOE and LARRY

```
$!
$! Common system disk startup procedure for LARRY and MOE
$! called by system command prodecure.  Created 31-Aug-1984.
$!
$ set noon
$
$ mount/system $255$dua2: cssework1 WORKA:
$!
$ define /system /exec common$disk sys$sysdevice
$ define /system /exec sys$sylogin sys$manager:sylogin
$ define /system sys$public sys$sysdevice:[public]
$ define /system /exec calendar$holiday sys$system:
$ define /system /exec calendar$data sys$login:
$!
$! Define the network logical names
$!
$ define /system /exec net$library sys$sysdevice:[netlib]
$ define /system /exec netlib sys$sysdevice:[netlib]
$ define /system /exec net$ sys$sysdevice:[decnet]
$ define /system /exec proxy10$ sys$sysdevice:[proxy10]
$!  Execute command procedure to set permanent terminal characteristics.
```

```
$!
$ @sys$manager:terminals
$!
$!  Start the various queues
$!
$ @sys$manager:startque
$!
$! INSTALL privileged and shared images
$!
$ run sys$system:install
basic /share /open
calendar /priv=sysprv
fortran          /open/shared
macro32 /share /open
pascal /share /open
runoff /share /open
$!
$!  Purge the operator's log file.
$!
$ purge/keep=2 sys$manager:operator.log
$!
$!
$! Install the VAX-11 Datatrieve images
$!
$  @sys$manager:dtrstup
$!
$!
$!  Startup VAX11 RSX
$!
$ @sys$manager:vax11rsx
$!
$! Definitions for VAXEln
$!
$ define /system /exec eln$ sys$sysdevice:[eln]
$!
$!
$!****************** misc assigns *********************
$ define/system   fal$log   "1"    !turn on logging
$!
$ EXIT
```

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

READER'S COMMENTS

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent:

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

Do Not Tear - Fold Here

Cut Along Dotted Line